DESIGNING A USER CONFIGURABLE ONLINE COMMUNITY FRAMEWORK

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon

By

MANJU SHREE CHAVA

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

ABSTRACT

Content Management Systems (CMSs) are widely used to create online communities supporting organizations, classes, and groups. These communities provide various functionalities, e.g. discussion forums, shared repositories for documents and links, collaborative spaces, and different communication channels, like chat or instant messaging. Often the range of functionalities offered is unnecessarily rich, and some remain unused, leading to cluttered users' workspaces and difficulties in finding information. Currently, communities that are developed with CMS do not allow user customization. Even for the community owner (e.g. a teacher, a group manager), it is hard to customize the functionality and interface of a community, because this requires some programming skills. I have designed new CMS allowing users of an online community (both owners and regular users) to design and configure their personal view of the community's dashboard by adding the functionalities that are present in the community's homepage and arranging them on the screen according to their preferences.

ACKNOWLEDGEMENTS

I am heartily thankful to my supervisor, Dr. Julita Vassileva, for her encouragement, guidance and support though out my research. I also thank her for offering me the opportunity to receive excellent education under her guidance.

I would like to thank my committee members, Dr. Kalyani Premkumar, Dr. Ralph Deters and Dr. Gord McCalla for their time, interest, and helpful comments.

My warmest gratitude to all my colleagues in the MADMUC Lab for their kind friendship and providing a fun learning environment. Furthermore, I am grateful to Jan Thompson for her kind help.

Lastly, I would like to thank my parents Mallikarjuna Rao Chava, Uma Kumari Chava and my brother Rakesh Chava for their support and love all the time and who simulated me to develop my career. I appreciate my husband Dr. Venkat A Kolla for his encouragement, unconditional love and support during my M.Sc. and finally my son Shashank Kolla, who made me enjoy my life and studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AIR	Adobe Integrated Runtime
CMS	Content Management System
FI	Full Interface
GUI	Graphical User Interface
HTML	HyperText Markup Language
LMS	Learning Management System
LCMS	Learning Content Management System
MADMUC	Multi-Agent Distributed Mobile and Ubiquitous Computing
MXML	Macromedia eXtensible Markup Language
MySQL	My Structured Query Language
PHP	Hypertext Preprocessor
PI	Personal Interface
RDBMS	Relational Database Management System
RIA	Rich Internet Application
SOA	Service Oriented Architecture
SWF	Shockwave Flash
UCCD	User Customizable Community Dashboard
WWW	World Wide Web
XML	eXtensible Markup Language

CHAPTER 1
INTRODUCTION

The World Wide Web (WWW) has been growing rapidly in recent years due to the proliferation of blogs authored by end-users, photo and video-sharing sites, online communities and social networks. The number of users using the internet has grown from 3.6 million in December, 2000 to 1.8 billion in December, 2009 (Miniwatts Marketing Group, 2010). We call these multi-user applications as "online communities". Online communities are places where users interact with each other for contributing and sharing content, rating and commenting. Examples of online communities are discussion forums, blogs, content sharing systems (for photos, videos etc), social networking sites etc. Online communities are proliferating, but few of them reach large scale. When social applications or online communities are built for a particular purpose and within an organization e.g. class support communities, or websites for networks of volunteers etc., Content Management Systems (CMSs) are typically used.

A CMS is defined as "*A collection of tools designed to allow the creation, modification, organization and removal of information from a website*" (Prayas, 2008).

CMSs enable non-programmers and designers to create powerful and professionally looking sites that support multiuser interactions like discussions, content sharing, commenting and rating. Currently there are over 1000 CMSs available (most of them are open source) on the WWW. Examples are Drupal, Moodle, Joomla, MediaWiki, ATutor, Movable Type, etc. There are different types of CMSs: Web CMSs, Enterprise CMSs, Document management systems, Learning Management Systems, etc. They reduce the amount of coding and ease the design of interface for web applications. Unlike html editors or web editor packages, however, CMSs

require some amount of programming knowledge from the developer. The web sites designed by each of these CMSs have the same look and feel (Myers et al., 2000). All the CMSs that exist today allow only the owner/developer to make changes/customization to the interface and add the required functionality. For example if a social site with discussion forum functionality is developed using a CMS, it may also have additional functionality added by the developer/owner, e.g. creating a feed for the discussion, rating, or tagging functionality. However, if the developer/owner has not added feeds functionality when designing the site, then the end-users of the forum will not be able to add it and consequently, will not be able to subscribe to a feed for the discussion, rate or tag posts. The end-users are restricted to interact with the site as designed by the developer/owner.

CMSs support four roles that can be played by users.

- A user in the role of a developer can design and develop the website.
- As owner, the user is in charge of that particular website and may have special rights to add and remove users, set policies for the users and content, as well as occasionally act as or call a designer to add functionality or change the interface of the community.
- A user in the role of an administrator manages the content that is present on the website.
- An end-user is a person who interacts with the content and functionality of the site.

These four main roles can overlap with each other and some users can have more than one role. For example the user in the role of an owner can act as an administrator for managing the content and setting the user policies.

Software applications are normally designed by professional software designers after a phase of thorough requirements elicitation and engineering. However CMSs allow online communities to be designed by non-professional software designers, and often the developed communities are maintained by the community owners/administrators who would be using some ad-hoc principles and their own experience to add new functionality that they feel is needed. For example, teachers designing a social site for their students in a given course use their own intuitions and the provided templates by the CMSs, but do not analyze formally the requirements of their students. Therefore, often the functionality and interface design are not optimal. It often happens that the application has features or functionalities which are not useful for a particular user. He/she may not want to see those functionalities all the time, since they clutter the interface and may be confusing. To deal with this problem, users should be given the opportunity to customize their interface to the application according to their wishes and arrange the functionalities they use and like the most. Just like most people are more satisfied to live in a house which they have designed according to their wishes and arranged all the things in whichever way they want, a regular user of an online community should have the possibility to customize the interface and functionality according to his or her convenience. So, it would be beneficial for CMS to support customizations by the user.

In my research work, I focused on developing a CMS called "*MUCCD (Manju User Customizable Community Dashboard)*" that allows end-users to customize the functionalities and

layout of their views without the use of programming. This would also ease the task of the community designer since he/she doesn't need to make crucial decisions about convenience and functionality.

In the MUCCD, the developer/owner who is creating the community website (for example, a teacher) is able to choose the required functionality for the community and arrange the Graphical User Interface (GUI) widgets implementing the functionality according to his/her wish. However, in contrast to other CMSs, MUCCD allows the end-user of the website to customize his/her interface to the community website, by creating their own community dashboard in which he/she can choose a subset from the functionalities that are provided by the owner/developer or use all the functionalities that are provided in the community. After choosing the functionalities, the user can resize the visual elements containing the functionalities, group them and place them on any location on the screen. The users cannot create new functionalities but only need to choose from the functionalities that are provided by the owner/developer. Even though the combination of functionalities from different communities in the MUCCD Dashboard resembles a mashup application, this application is not a mashup. Typically, mashups combine functionalities from independent providers or different communities which are not linked or related to each other. In this case the functionalities in the personalized dashboard are chosen from communities that are created by using the MUCCD CMS, not from independent providers.

This CMS was implemented as a proof of concept and evaluated in a small case study to test the hypothesis that users will actually do customizations when given the opportunity and that the current design supports them well in the process of customization. This CMS was used to develop several different websites for online communities. In the study, users were observed and

data was collected on whether users make customizations, what kind of customizations are popular and how usable the customization options are. The results confirmed that users liked the idea to access multiple communities at the same time from a dashboard application and that they were able to customize the functionality and interface of the dashboard according to their preferences. The results also provided some new insights in the kind of customizations that users tended to do in the context of their online community dashboard. (In this thesis "customizable interface" is used as an alternative to "adaptable interface").

The remainder of the thesis is organized as follows. Chapter 2 provides a review of the most popular existing CMSs, on different customization approaches and strategies for doing customizations. Chapter 3 presents the design and implementation of a user-customizable community interface. Chapter 4 describes the evaluation approach which consists of: the pilot study, the main study and the results obtained from them. Chapter 5 presents the conclusion and areas for future work.

CHAPTER 2
BACKGROUND AND RELATED WORK

This chapter presents a review of related literature. It starts with a brief introduction of some CMSs that are used for creating online communities. Different types of customizations that can be done on web applications i.e. graphical user interface and content customizations along with different approaches for doing customizations – adaptive, adaptable and mixed-initiative, depending on who has control over the customization process. These approaches give information of how the user or system does the customization of interface. Later possible strategies for doing customization are described, depending on at what point of time users choose to do the customization. This chapter ends with a summary of customization approaches and discussion on which of them are more appropriate to be applied in CMS.

2.1 Content Management Systems

Updating a website developed using traditional web-design tools like Dreamweaver or FrontPage is difficult (Robertson, 2003). The developer may have lost track of all the pages that are present on the site or someone else may have developed the website and its structure has become convoluted. It is also difficult to get back to the information that was present on the site on a particular day of a particular year, since generally websites have no version management tools.

To solve these problems Content Management System (CMS) is used which is web software which allows application developers to create websites without using advanced programming skills. CMSs separate the content from the presentation. The content of web pages can be updated easily and new pages can be added to the website without any difficulty. The

content can include graphics, images, links, videos, music or documents. The appearance and the structure of a website, like where the pages go and how different pages are linked to each other can also be managed easily.

For designing websites using CMS, developers do not require knowledge of HTML. Maintaining a website developed by using CMS is also easy by providing authorization to different people. Using CMS, anyone can create, maintain and update a website whenever needed (Fenton, 2008). There are over 1000 CMSs available on the WWW and choosing an appropriate CMS is an important decision for any organization (Williams, 2009).

Web 2.0 has made the web more dynamic allowing conversations and content contribution by end-users. Some examples of web 2.0 applications are blogs, Really Simple Syndication (RSS), wiki, forums and social networking sites. CMSs allow creating dynamic sites that support end user participation and contribution. There are CMSs which are used only to develop such applications. Examples are MediaWiki for wikis, Moodle and Blackboard for education, WordPress for blogs and Drupal for designing any type of websites (Shannon, 2009). CMSs also provide interface templates which are pleasing to use.

Common features of CMSs are (Arakelyan, 2009):

1) *Easy content editing:* Web editor tools with multi language support are provided by CMSs for updating and editing content at anytime.

2) *Preview before publishing:* CMSs allow developers to preview content before publishing to make sure that the content is in the correct place or requires any further changes.

3) *Extendable functionality:* Sometimes developers of CMSs may not be satisfied with the functionalities they have. For this CMSs allow extensions to be added by third party providers.

4) *Templates:* CMSs provide developers with different templates and allow them to choose from those templates for designing the website.

5) *Version Management:* CMSs allow users to create content and maintain different versions of the content so that they can be accessed at later time.

6) *Different User Levels:* Authentication can be set to different users for accessing content.

7) *Automatic system updates:* In order to keep a system up-to-date with the latest updates and new functionalities automatic system updates are provided by CMSs.

8) *Searching:* Finding the functionalities which developers need in designing a website can be done by using search engines that are provided by CMSs.

There are both advantages and disadvantages in using CMS.

Advantages of using CMS:

- Updating a website by using a CMS is easy because the design and the content part are maintained separately in the CMS.
- Even novice users can create a website with functionality and layout similar to a website created by an expert.
- Content remains the same even if the design changes.

Disadvantages of using CMS:

- Changes to the design of a website are restricted as provided by a CMS and cannot be done easily unless the developer is an expert.
- When designing a website using CMS, different pages cannot have different templates or different cascading style sheets.
- For smaller and simpler sites, CMS is probably an overkill.

CMSs are becoming more popular as tools for creating online communities because they bring people together either for socializing or for working together. Hinchcliffe (2008) summarizes some of the most commonly used CMSs to create online communities. Next a brief introduction of these CMSs is presented.

Joomla: (http://www.joomla.org)

Joomla which means "all together" in Arabic was released on September 16, 2005 which was a re-branded release of Mambo 4.5.2.3. Joomla is used to develop personal homepages, corporate web sites, social network, eCommerce and archive sites. It is an open source CMS which is easy to use and install. It is developed using PHP and MySQL. Using Joomla a web designer or a developer can easily develop a website for end users. If an end user needs any other functionality, then there are thousands of functionalities for extensions available on the Joomla Extension Directory.

In Joomla there are over 200,000 community users and contributors and the community is growing quickly. It has an official discussion forum with over 1.3 million posts. The main disadvantage of this CMS is that it has a steep learning curve (Tradocaj, 2009) and changing the layout is difficult with the bulky CSS and JavaScript.

Drupal**:** (http://www.drupal.org)

Drupal is a free open source CMS developed using PHP. Drupal is one of the most powerful CMSs and has a vast number of modules. Many people use it to develop different kinds of websites like ecommerce, blogs, forums, social networking sites and picture galleries. It has an active developer community. Different modules can be selected depending on the template that is used.

Drupal provides many different themes which can be chosen to customize the look and feel of a designed website. It supports 45 different languages for creating and modifying the content. The disadvantage of this CMS is that it is difficult to learn and needs frequent security upgrades (Karen, 2009).

PHP-Nuke: (http://phpnuke.org)

PHP-Nuke is based on PHP and MySQL and is used as an automated news publishing website. The current version of PHP-Nuke is 8.1 and is available after pay but the versions before 7.5 are free to use. PHP-Nuke is available under GNU (General Public License) and the purchaser of the software is free to distribute the source code after purchase of the product.

In websites created by PHP-Nuke, users and editors can post news articles and only registered users can comment on the articles. Some of the standard modules that are present in PHP-Nuke are forums, search, submit news, advertising, surveys, topics and web links. The problem with PHP-Nuke is that it has some security issues regarding PHP code and SQL.

Zikula: (http://zikula.org)

Zikula is a free open source CMS which is easy to use, secure, flexible and has high performance. Websites related to blogs, portals, companies, shops, forums, etc., can be created using Zikula. Zikula uses PHP and is compatible with MySQL, PostgreSQL and Oracle.

If developers are not satisfied with the functionalities that are present in the CMS, they have the ability to extend the third party modules. Simple HTML is used for displaying the content on a website, it allows the site administrator or the developer to change the look and feel of the website and arrange the content. There are many built in themes that can be chosen by the

developer. Zikula has forums support which provides answers to questions in different languages.

SharePoint Community Portal: (http://sharepoint.microsoft.com/sharepoint/cks/Default.aspx)

A community website can be created for any group of people with a common interest using the SharePoint technology. Websites can be hosted using Microsoft's SharePoint which can access the documents, information stores, shared workspaces, wikis and blogs. Community kits are provided by SharePoint with tools, templates, source code which allows users to create a community website based on the SharePoint technology. The advantage of the websites designed with this portal is that the team members don't need to install the appropriate program to read the document.

Lithium: (http://lithium.com)

Promoting the company through customers is important way of advertisement. Customers can do marketing by word-of-mouth which is a more persuasive way of advertisement since it is personal. Customers can also help each other (peer-to-peer support) which reduce the costs for support that the company needs to pay. New ideas can also be obtained from the customer base. A company forum with Lithium's "social web connect" can provide solutions to companies who want to be connected to their customer base.

Strong customer relationships can be built by integrating online customer communities, social networks and existing customer relationship management systems. By creating a customer network on the web, customers promote a company, innovate on the services provided by the company by giving new ideas and supporting each other. With the customer network that is available for a company, profits can be increased.

DotNetNuke: (http://www.dotnetnuke.com)

DotNetNuke a web CMS by DotNetNuke Corp. is used for the developing application using Microsoft VB.NET. It was established in 2002 and is an open source web content management system. DotNetNuke originated in another project called "IBuySpy Workshop". It provides modules like blogs, forums, wiki, photo galleries, and mailing list. Beyond these additional modules can be obtained from third-party vendors and users in the community. The look and feel of an application can be customized by the owner/developer using the skins that are provided.

DotNetNuke offers two editions - Professional Edition for business purposes and a Community Edition for developing communities. Some of the features of DotNetNuke CMS are that it is easy to use and install, powerful in case of supporting multiple websites from a single application installation. It is localized in the sense that it has multi-language support allowing the administrator to translate the website into any language.

Community Server: (http://telligent.com)

Community Server which is now called "the Telligent Community" was created in 2004. It was originally merged from three open source ASP.NET projects, namely the .Text blog engine, nGallery photo gallery and the ASP.NET Forums. Telligent is a community and a collaboration software product by Telligent Systems. It is built with C#.NET and the current version that is available is Telligent Community 5.0. The core applications that are present in this software are blogs, forums, wikis and media galleries. Functionalities from third parties can also be integrated with the web application that is developed.

The look and feel of an application can be managed by using the customizable and flexible user interfaces. Using Telligent Community, small communities can be created which

are targeted to particular customer or some private groups of users. Discussion forums are available for the community where the user's questions can be answered (Wikipedia, Telligent Community, 2009).

KickApps: (http://www.kickapps.com)

KickApps can be used to create social networking sites, photo and video sharing sites where the developers can create their own widgets, create custom video players, etc. A social network can be created by choosing the functionalities which the developer wants. Some of them are the ability to add friends, participate in different groups, send private messages to friends, customize the users' profile and many others. Photo and video sharing is also easy and the users of the website can rate, comment on the articles.

Jive Software: (http://www.jivesoftware.com)

Jive Software is a Portland software company founded in 2001. By using Jive Social Business Software (SBS), collaboration, community and social network software can be developed. It offers various modules, for example a bridging module which can connect an employee community with the customers and partners' communities. The video module enables users to add video, image and audio files to the website. Other functionalities that are provided are social bookmarking, file uploads, search results redesign and profile tooltips.

2.2 Learning Management Systems

A special class of CMS is targeted at creating and managing educational content and interactions. A learning management system (LMS) is a "*software application for the*

administration, documentation, tracking, and reporting of training programs, classroom and online events, e-learning programs, and training content" (Ellis, 2009).

A LMS provides the instructor the ability to create and deliver learning content to the students, monitor the student participation, and to access the student performance. LMSs enable students to access the learning content from anywhere and at anytime though web, to upload assignments, take on-line exams, discuss course-related topics. They allow access of users in different roles: instructors, students, markers, etc. A variety of LMSs is used in educational institutions and in companies to train employees using the e-learning and e-certification. These LMSs come in different languages.

Edutech Wiki (2010) lists the following typical components present in LMSs.

a) Course Management which lists the courses, registration for courses, credit information and syllabus, and the pre-requisites that are required.

b) Teaching materials for each course

c) Self-assessment quizzes

d) Synchronous communication like chatting, teleconferencing.

e) Asynchronous communication through emails and forums.

f) Student tools for self tests, bookmarks, progress tracking etc.

Another term that can be found in literature is "Learning Content Management System" (LCMS). This is "*a related technology to the learning management system (e.g., Murray Goldberg's WebCT), in that it is focused on the development, management and publishing of the*

content that will typically be delivered via an LMS" (Wikipedia, Learning Management Systems, 2010).

LMS and LCMS are different in that using a LMS we cannot create and manipulate courses and cannot reuse the content of one course to build another course. A LCMS is a multi-user environment focused on content-creation and management, where developers can create, store, reuse, manage and deliver the content. Despite of this difference, the term "LMS" is nearly always used in place of both LMS and LCMS.

Some examples of LMSs are listed below.

Moodle: **(**www.Moodle.org**)**

The acronym of Moodle is Modular Object-Oriented Dynamic Learning Environment. This is a free open source LMS used by instructors to create online learning communities. It has over 50 thousand registered sites and provides rich interaction for online courses. Moodle uses PHP and can run on Windows, Mac OS, Linux and UNIX. The databases used are MySQL or PostgreSQL. Since it is an open source platform, people can add additional features.

Moodle was created by Martin Dougiamas, a WebCT administrator at Curtin University, Australia. His Ph.D. research regarding the open source software for teaching and learning has influenced the design of Moodle (Wikipedia, Moodle, 2010). Moodle has a feature for allowing students along with teachers to contribute information to the course web site by either adding the comments to the topics or by collaboratively working on wiki topic. Some of the features that are present in Moodle are forums, blogs, wikis, surveys, chat, glossaries, quizzes with different kinds of questions etc.

WebCT and Blackboard: (http://www.blackboard.com/)

WebCT is the world's first widely successful course management system for higher education. At its height, it was in use by over 10 million students in 80 countries. It was acquired by Blackboard and is now a part of their BlackBoard Learning System. According to Wikipedia:

Blackboard Learning System ... is an online proprietary virtual learning environment system that is sold to colleges and other institutions and used in many campuses for e-learning. To their WebCT courses, instructors can add such tools as discussion boards, mail systems and live chat, along with content including documents and web pages. The latest versions of this software are now called Webcourses (Wikipedia, WebCT, 2010).

With Blackboard resources can be accessed at any time during the day and students can have access to any type of resources from the coursework to extracurricular activities. Accessing all the resources can be done with one login which saves time from having multiple logins to different systems which is time-consuming. Blackboard also allows instructors to keep track of students' behaviour, posts that are added to the discussion forum, things to be done and perform the actions accordingly. Blackboard technology helps to prevent plagiarism without leaving the course environment. Working in groups is also popular in a university environment. By using Blackboard students can create a personalized interface for the group by arranging the items within a course and assign tasks to individual members of the group.

ATutor: (http://www.atutor.ca/)

It is an open source learning LMS used by teachers and students to create an affective learning environment with a vast number of features. Administrators can install ATutor and can

import functionalities that are provided by the application and can create their own templates. Here the instructors have ability to choose the tools and modules they want for their students to use and place them on their course home page. Both instructors and students can manage the courses that they have registered for. The students can discuss their projects through the forums and can collaboratively work on their projects. The navigation in ATutor is very simple by using text and icons. ATutor uses PHP and can run on Windows, Linux and UNIX. The database used is MySQL.

All the CMSs and LMSs discussed above share one feature: once designed, the websites created with these CMS cannot be customized by the end-user to fit the preferences of the end-user. They have "one interface for all". This is not problematic in simple sites, offering a limited set of functionalities and designed with a clear view of the user population that will access them. However, the complexity of sites that support communities increases dramatically, since new functionalities become available all the time and designers/owners are tempted to add them to their sites. The resulting interfaces can become very complex and the need to provide ways for user customization arises. In the next section, the focus shifts to the customization of user interfaces. Different approaches for customizing interfaces are presented.

2.2 Customization

Customization of web applications is the main aim of my research project. Normally, the applications developed by using the CMSs contain a wide range of features or functionalities including some which are not useful for a particular user and clutter the user interface. For this users should be given a chance to customize the application according to their interests.

Customization of applications can be done in two forms: Graphical User Interface (GUI) customization and Content customization (Bunt, 2007). In GUIs, users interact with interfaces using graphical elements like menu bars, toolbars and buttons. These graphical elements are used to trigger the application functionalities. GUI customization involves the graphical elements in order to adapt the interface according to the user's choices. In contrast to GUI customization that is done on the interface elements related to application functionalities, content customization is done on the information that is manipulated through the application's functionalities. An example of content customization is arranging the results obtained from a search widget based on the content ratings. Both types of customizations have challenges and issues. For content customization, content will always be changing since generally more content will be added to the systems. For users to access the right content they should have an option to customize the content and adapt the application functionality (i.e. search) according to their choice. In the case of GUI customization, changes to the GUI will be done rarely and so a user can expect that the GUI will remain static (Bunt, 2007). Regardless of this static nature of the GUI, users mostly prefer to have GUI customization than content customization. This is because by arranging the graphical interface elements, users can become familiar with the interface. With practice users can remember where the graphical elements are present in the application and can access the application more easily (Norman, 1991).

2.2.1 Types of Content Customization

The presentation of content to the user depends on *what* content will be presented, *how* it should be presented and *when* it will be presented. Along the dimension of *what* content will be presented, approaches for customization have addressed recommendations of the customized products (Chen and Pu, 2007), selection of topic in adaptive web site (Papanikolaou et al., 2003),

providing hints in intelligent tutoring system (Bunt et al., 2001). Along the dimension of *how* the content is presented, customization approaches can address the structure, layout and modality used (Zhou et al., 2005). Customization of *when* the content is presented to the user can also be done for example, at the beginning or as the task progresses (Horvitz et al., 1998; Jameson and Schwarzkopf, 2002).

2.2.2 Types of GUI Customization

Bunt (Bunt, 2007) has proposed three ways in which GUI can be customized.

- Macro definition
- Feature management
- Cosmetic customization.

These three types differ in the following categories:

- The amount of effort the user puts in order to customize the interface.
- The size of the interface that needs to be customized.
- The impact of customization on users' performance after customizing the interface.

In macro definition, if there are any frequently used sequences of commands for an application, then those particular functions can be added as menu items or toolbar items or buttons so that they can be accessed easily. Examples for macro definition can be found in MacLean et al., (1990), Oppermann, (1994), Page et al., (1996). An example of sequence of commands for a word-processing application in WordPerfect 6.0a for Windows is setting the Font with New Times Roman, Size with 12pt, and Bold (Bunt, 2007).

Feature management is related to customizing the functions that are present in the existing interface. An example of this is, if there are many frequently used functions in the menu

bar then those functions can be placed on the toolbar so they can be easily accessed (Page et al., 1996). In this case the functions are being duplicated, but in order to avoid duplication of the functions, the set of functions that are accessed more frequently can be placed in an order so that these functions are on the top of the list or allowing only these functions to be present in the list.

Lastly cosmetic customization is related to simple customizations that are done on the interface i.e. moving the buttons, zooming, changing the color of items. Examples of this kind of customization can be found in Page et al., (1996), Gant and Nardi, (1992), Jorgensen and Sauer, (1990).

My research is related to GUI customization. From the three different types of GUI customization, feature management is the one related to my research. The interface should allow the user to have easy access to the most frequently used functions. Less frequently used functions should not have impact on the user's ability to access the most frequently used functions. Macros can also be used for doing the customization but the impact of macros on the interface is much smaller. This is because macros allow users to group few functions to form a single function. Macros reduce the number of functions which the user searches but increases the interface complexity.

2.3 Approaches for GUI Customization

Customization allows design flexibility. Flexibility, according to Oppermann (1994), is to give freedom for each specific user of the application in designing and choosing the tasks for increasing their efficiency in using the application. With the flexibility that is provided by the

application, designers have fewer tasks in designing the application according to the user's interest and user alone can make the decisions on how to perform a task.

There are three approaches for GUI customization.

1) Adaptive: where the system does the customization
2) Adaptable: where the user does the customization
3) Mixed-initiative: here both the system and user cooperate to do the customization.

The next sections present briefly these three approaches.

2.3.1 Adaptable Approach

Adaptable interfaces are used in many commercial applications which allow users to do GUI customization. Many existing applications have adaptable interfaces, for example, Yahoo, iGoogle, Excite sites. Research done on the adaptable interfaces show the amount and type of customizations users do on these interfaces.

Do users customize?

Customizable interfaces require the user to be familiar with the mechanism for customization and to spend a lot of time for learning and doing the customization. As a result only a few users customize their interfaces. Surprisingly the results of an experiment by Page et al (Page et al., 1996) showed that 92% of participants from a field study with 101 participants did customization in a word-processing application (WordPerfect 6.0a for Windows). The customization involved creating short-cut menus and adding new interface items.

Jorgensen and Sauer (1990) conducted experiments in three different contexts: 1) the "IBM Assistant", 2) a business application package and 3) an unidentified operation system. The

experiment was done with 10, 720, 27 participants in different contexts. The authors found that almost 50% of the users did at least some type of cosmetic customization.

Mackay (1991) did two customization studies on the UNIX software with 69 participants (one with 51 participants and other with 18 participants which included managers, secretaries, technical and non-technical staff) and found that 78% of the participants did some sort of customization. However, she observed that users were not using the customization facilities even though they could have benefited from them. She proposed that users did the customization because 1) they had some patterns which were repeated and wanted to match those patterns by doing the customization, 2) they wanted to stop or remove something that was disturbing them and 3) they wanted to make the new software similar to the older one which they were more familiar with, to avoid confusion. Most of the users thought that there was not much time to customize the software and that the software was complex.

A new form of customization is present in the World Wide Web, where the users are given option to personalize the web pages related to the portal sites. An example of this is Yahoo where the user can browse or search for information they need. For this portal site, users have option to select some kinds of news information in which they are interested (e.g. weather, sports, finance, technews, gossip, etc.), and various functionalities, for example, a horoscope, calendar, etc. and set up a personal page containing all of these, which is called My Yahoo. This personal page can be used as the main entry page to the Yahoo repository rather than the main entry page.

The use of My Yahoo is described by Manber et al., (2000). They described that the majority of Yahoo users used the default personal page that was provided and did not customize

their My Yahoo page. The reasons for not doing the customizations were 1) the default entry page was good, 2) the customization tools are difficult to use and 3) users did not need the complex personalization. The authors surmised that the main reason was a combination of all the three reasons. They found that people did not understand the concept of customization and the benefits obtained from doing the customization. For the users to perform the customization, the tools must by easier to use and should be also available for the less-experienced users.

There are many applications with adaptable interfaces (e.g. Microsoft Word 2000 by McGrenere et al., 2007, WordPerfect 6.0 by Page et al., 1996, User Interface Facades by Stuerzlinger et al., 2007) that allow users to add or remove functions from the menu bar or toolbar and to move functions from menu bar to the toolbar or vice versa. Even though these types of customization facilities are available, there is little research in designing applications using these facilities (McGrenere et al., 2007). The adaptable interfaces takes lot of time to learn and the adaptations are done mostly by experienced users.

Two-interface Model

To make the customization more attractive and simple to the user, McGrenere and Moore (2000) proposed two-interface design in an experiment with MSWord97. This two-interface design is an adaptable model where users had the option to choose between the full interface and the personalized interface. The default full interface had all the functionalities and the personalized interface had only the functionalities which users chose. Users had an option to toggle between the default and the personalized interface using a toggle button. By default, in order to make the application simple to use, the personal interface was launched in the application when the user logged into the system. This experiment was evaluated with 53 users.

Users liked the system with the two interfaces – the customized and the full interface, and they switched back and forth between them when needed extra functionality. McGrenere and Moore found that the majority of users did not like the word processor with only the functions needed for performing a task because they wanted to discover new functions. Some users had negative experiences with the full interface of the application due to the large number of functionalities present in it.

McGrenere et al. (2002) proposed a two-interface model with MSWord 2000 which has a full and a personalized interface. This model with the customizable mechanism was evaluated with 20 participants for about six weeks. The authors found that the users liked the two interface model and about 70% of the users spend most of their time on their personalized interface. This study also showed that users deployed different approaches to do the customization. 32% of them did most of the customization at the beginning of the study and the remaining 68% did the customization in an incremental manner. 37% of the users added all the functionalities and the rest added the functionalities in incremental steps and only the ones which they needed.

The studies discussed above showed how users added or removed functionalities and why they have added these functionalities; they focus on strategies used for customization, but they did not provide much information of how the customized interface affected the performance of the users with the system.

Level-structured or Layered Interfaces

When there are users with different experience levels who are required to use one system, a level-structured approach is used (Shneiderman, 1997). Level-structured approaches are sometimes also called “layered approach” or “spiral approach”. Shneiderman says that a level-

structured approach should be used to make the system easy for the novice users so that they can make the correct choices with few mistakes. The main idea is to block certain functionalities for users with lower level of experience, so the allowed functionalities are organized in hierarchical layers starting from a small set at the lowest layer, and increasing the set of functionalities at each next layer up the hierarchy.

A classical level-structured design has two or more interfaces representing different functionalities. If a user is working at a level with few functionalities and wants to have another functionality which is in another level, then she needs to move to the next level which has more functionalities. A number of commercial applications have level-structured interfaces e.g. Hypercard and Framemaker (McGrenere et al., 2002). Another example is Eudora, the now nearly extinct, but previously very popular e-mail client used on Macintosh and Windows operating systems, which offered the level-structured approach across their two versions – Pro (paid) and Lite (free). This level-structured type of interface leads to frustration in Eudora users because the user is forced to move to the next level (McGrenere, 2002) to be able to use the richer functionality.

Another example of level-structured approach is Carroll and Carrithers' Training Wheels interface (1984), which was used for an early word-processor. This system had only one interface and the functionality which was not needed for doing simple tasks was blocked. When the user tried to unblock these functionalities, a dialog box popped up showing that these functionalities were not available in this version. An alternative system with the full functionality was made available for users with sufficient experience, i.e. with the increasing experience of the users, the "training wheels" were removed and they had access to the full version. Carroll and

Carrithers found that novice users identified and completed the tasks more easily in the first version than in the second one.

Shneiderman (2003) and Christiernin et al., (2004) proposed another type of interfaces called layered interfaces where the application is divided into number of layers or interfaces. In layered interfaces users can switch the interface either by changing his/ her expertise level or depending on their tasks. This type of customization is called "user-controlled GUI customization". Designing this type of layered interface is difficult as the number of functionalities increases and it is also difficult to design an interface for different types of users. These types of interfaces are not evaluated much (Bunt, 2007). The results from the comparison of two-interface model with full-featured interface proposed by Findlater and McGrenere (2007) showed positive impacts on the two-interface model.

Another adaptable interface is provided by Stuerzlinger et al., (2007) in Facades, where the users were allowed to reconstruct the application in a separate window. They could change the layout and all other functionalities that are present in the interface. Though this is a powerful mechanism it is yet be evaluated.

In adaptable interfaces, the user has full control of the interface. Even though users like to do the customization, they rarely customize the application because it requires users to learn the customization options that are provided to them and takes time for doing the customization (McGrenere et al., 2007). Again some of the users may not know that they can customize the application and some may not know all the functionalities (Bunt et al., 2004).

2.3.2 Adaptive Approach

Contrary to adaptable interfaces where the user can control the customization of interface, in adaptive interfaces the system automatically does the customization based on the user's needs and actions. The most popular and commercial example to adaptive interface is smart menus in MSWord 2000. Here the user is provided with a small menu which has only few functions that were recently accessed or the most frequently used ones as opposed to the full menu which has all the functions (Bunt, 2007). Users can access all the items by using the full menu. But the disadvantage with this is that when the short menu is expanded into the full menu new functions combine with old ones in the smaller menu and users need to re-scan the complete menu to find the desired function. Other examples of adaptive interfaces are Windows XP Start Menu and MS Word where the menu adaption will be done based on the usage of the functions by the user (Gajos et al., 2006).

Another example of adaptive interface is Adaptive Telephone Directory (Greenberg and Witten, 1985) which has names organized hierarchically. This telephone directory contains 2611 names and these names can be adapted according to the user's interaction so that the most frequently used names are placed on the top of the hierarchy and the less frequently used names are placed in the bottom of the hierarchy. An experiment was conducted with 26 participants to check whether they liked the adaptive interface for organization of names depending on their use or the static interface where the hierarchy remained constant. The results of this experiment showed that 69% of the participants liked the adaptive interface because it reduces the effort for searching the desired name.

Some other examples of adaptive interfaces are Adaptive Prompting (Malinowski, 1993) interface which has a menu that contains functions that are most likely used and the most appropriate ones based on the user's context. The systems AIDA (Cote-Munoz, 1993) and Skill Adaptive Interface (Gong and Salrendy, 1995) adjusted the functionality in such a way that it was presented to the user either by using the GUI or command line interface. In the adaptive interface developed by Gong and Salrendy (1995), users interface changed from the menu driven interface to the command line interface after they used the menu item for certain number of times. The system notifies the user to use the menu item from the command line and then the menu item disappears from the list to a hidden portion of the interface.

An example of adaptive information hiding was used in the institutional hypermedia system Hynecosum (Vassileva, 1996). The aim of this application was to allow the institutional employees to have personalized access to the data that was relevant to their job-related tasks and rank. This was implemented by hiding the task-irrelevant hyper-links from the users (both from the index and from the local node/page). By organizing the user interface around the tasks they perform in hierarchical task menus, the access to the data required navigating the task hierarchy until elementary tasks were reached that provided access to the data. Novice users had to navigate down the task hierarchy (go deeper into the sub-menus) until they reached a level from which access to the data was allowed. This level depended on their experience with the system and climbed up to the root of the hierarchy for very experienced users, who did not have to navigate through the tasks but had direct access to the data. The main assumption underlying this approach was that novice users know their tasks better than they know how the data is structured in the system, but with their learning of the system, they learn about the data-organization and they would be able to find the data they need by just browsing the data organization (hypertext)

directly. The evaluation of the system confirmed that the assumption was correct and showed performance improvement in terms of speed of finding the required information both for experienced users and for novices.

Split Menus

An example of adaptive menus which use positioning of items is described by Sears and Shneiderman (1994) where the concept of split menus was introduced. Most frequently used items were presented in the top part of the split menu and the less frequently used items are presented in the bottom part. This experiment was evaluated with 38 participants and showed the advantages of using the split menu. The experiment did not test whether the most frequently used items need to be at the top or bottom and items in the top menu were fixed before start of the experiment.

Findlater and McGrenere (2004) conducted a laboratory study to test the performance with 27 participants on Sears and Shneiderman split menus (Sears and Shneiderman, 1994) to test the static, adaptive, and adaptable menus. The split menus were of three variants: one with static menu, a second one, where the top half of the split menu was adaptable by the user, and a third one, where the system would adapt the functions based on the functions' usage frequency. Participants were asked to select a sequence of menu functions for each condition and those functions were presented twice.

An adaptive GUI customization which was unfortunately not evaluated can be found in Miah's experiment (Miah et al., 1997). Here the toolbar items and the entire toolbar are added and removed automatically based on the use of the toolbar items and the toolbar from the time

they were created. So from these examples we can say that maintaining a history of items that are accessed is important to adapt the functionalities according to the user's needs and actions.

An example of adaptive interfaces is found in search engines and web browsers which have an "autocomplete" feature used to predict the word or phrase that is being typed without the user actually typing it completely. In web browsers, auto-completion of web addresses is a convenient feature which does not require users to remember the long and full addresses. In search engines, this feature suggests the queries as the user types in the query (Wikipedia, Autocomplete, 2010).

The adaptive interfaces are more focused on the technology for adaptation and evaluating these interfaces is more complex than the standard or adaptable interfaces. An example is the Eager system which highlighted the menus and objects on the screen to indicate what the user would do next (Cypher, 1991). Some adaptive interfaces are easy to use and are pleasing while others are confusing and frustrating for the user. The disadvantage of the adaptive approach is that the users can feel that they are not in control of the system (Dieterich et al., 1993; Fischer, 1993) and they may not understand or want the functions which the system is adapting (Bunt et al., 2004).

2.3.3 Mixed-Initiative Approach

The mixed-initiative approach is a combination of adaptable and adaptive approaches and was first discussed by Fischer (1993). A system with mixed-initiative approach is one which supports automatic adaptation based on the users' actions and needs, but leaves it to the users to do customization and just provides suggestions or hints (Horvitz et al., 1988). Due to the

disadvantages of the purely adaptive approach discussed in the previous section, most currently existing adaptive systems are mixed-initiative systems.

An example of a mixed initiative approach is EAGER (Cypher, 1991) which monitors the user's repetitive actions in a word-processing application, identifies particular patterns which are performed repetitively and suggests to install a macro for that operation. In order to make the user more confident in the need to make a customization, it highlights the next usage of this pattern in green indicating that it has correctly identified the pattern. When the user clicks on the new macro, EAGER automatically performs the steps and installs the macro. A study conducted with 7 users in a laboratory setting found that the users liked this way of pattern selection and they also did not like giving up control to the system. The paper, however, did not clearly specify if the changes made to the interface by installing a macro were permanent or not and whether the macros were available at a later time.

The mixed initiative system proposed by Benyon (1993) suggests to the users to use either a command line interface or a menu driven interface based on a model of the user's previous computer experience and types of errors they make. Clark and Matthews (2005) proposed a layered interface, which allows users to select the layers they want. Some layers are made visible to the user based on the documents the user has edited in the past. There are no evaluations of these two systems so there is not much information about their success or failure.

FlexExcel project (Krogsaeter et al., 1994; Oppermann, 1994; Thomas and Krogsaeter, 1993) is an extension of Excel software which implements a mixed-initiative customization approach. If any of the functions is used repeatedly, the system would make a suggestion for creating a new menu function or creating a short cut for that particular function. The notification

for available suggestions is a “Tip” icon which blinks three times and makes a sound. This system was tested with 13 participants and the authors found that the users had difficulty doing the customizations following the suggestions. The problem could have been either because the user had difficulty in understanding the suggestions in the way the system presented them or because there was some problem with the adaptive algorithm.

Debevc et al., (1996) proposed an Adaptive Bar which is used to manage the functions that are present in the toolbar of MSWord. In this system users can add or remove the functions from the toolbar with the help of a suggestion given by the system based on the frequency of use of those functions. The system notifies the user that it has suggestions by changing the color of toolbar and by producing a sound. The toolbar also displays the frequency of use of each item by changing its size. Evaluation of the system was done by comparing the Adaptive Bar with the MSWord toolbar. This experiment was conducted with 16 participants in a laboratory setting and they found improvement in user performance by using the mixed-initiative approach in one of the experiments. The performance was measured in terms of customization time required to do the experiment. The results showed that novice users used more of the mixed-initiative approach than the adaptable approach and with the expert users it was vice-versa.

In summary, some applications, for example, spreadsheets, image editing software, or word processors, contain many functions which are hidden in the menus. These functions are not useful for each and every user all the time. So the interface may appear to be cluttered and confusing. Customization of the interface using a mixed initiative approach can provide a good solution (Bunt et al., 2007).

Bunt et al. (Bunt et al., 2007) described a system called MICA (Mixed Initiative Customization Assistance), which was designed to combine all the advantages of adaptable and adaptive systems. In the adaptable approach, users are given full control of the interface and can customize the interface. In the case of the adaptive approach, the system will help users to customize by giving suggestions i.e. in a mixed-initiative way. Though these customizations help users to save time, they interfere with users' actions. The customization suggestions by these systems are based on the rules that were set to increase user performance and reduce the number of interruptions. This system does not interrupt the user but only gives subtle suggestions and the user chooses whether or not to customize the interface.

All three customization approaches have advantages and disadvantages. Fischer (2001) argues that the advantage of adaptable interfaces is that they put the users in control, and users know the task better than the automatic reasoning done by adaptive interfaces for customizing the interface. Adaptable interfaces might require more effort from the users, but users prefer them over the adaptive approaches or mixed-initiative approaches, because they do not understand how the automatic reasoning works and have problem trusting the suggestions.

2.4 Strategies for Customization

Bunt (2007) in her thesis proposed customization features for MSWord (2003) to prove whether the customization is worth doing i.e. whether the customization can improve the performance and whether the users need any help in doing the customization efficiently. She conducted two experiments. The first one aims to see if there is any chance of saving time in doing customization and the second - to see what features the users added to their personal interface and the strategies they used.

The strategies that users apply while adding the features to the personal interface are:

1) *No customization:* Here the user will use an interface containing all the features that are provided to him/her and will not do any customization.
2) *Up Front:* If a task is given to the user as a part of the experiment, then the user adds all the necessary features he/she thinks might be useful in completing the task.
3) *As You Go:* The user adds the features one at a time in a particular order as they are needed when doing the task.

The first experiment was conducted on GLEAN simulation (Kieras et al., 1995) with the three customization strategies, two tasks and each of four expertise categories (Novice, Intermediate, Expert and Extreme Expert). The two tasks were writing a letter and writing a report which have different levels of complexity and which are done by the users almost every day. The result of this experiment shows that, mainly for Novice and Intermediate users, customization is effective in terms of saving time. Up Front strategy is the best strategy for a user to save time.

The first experiment focused on the *when* dimension to see when the features are added to the interface for completing the task and the second experiment focused on the *what* dimension for the two customization strategies (Up Front and As you Go) to see what features are added to the personal interface i.e. all the features that are present or only the frequently used features. The results of this experiment show that when infrequently used tasks are also added to the system along with the frequently used ones, then the performance (in terms of amount of tome users spent on customization) of novice users decreases. McGrenere et al., (2007) showed that users rarely removed items from their personal interface. In order to help users to add or

remove functions from the interface, Bunt (2007) proposed adaptive support to help users in choosing the items in her MICA system.

2.5 Summary

The literature survey shows that there are different CMSs available for developing online communities. These CMSs allow the developer/owner of the community to choose the functionalities and interface that provides access to these functionalities by using the templates that are provided. But none of the discussed CMSs allow the user of the community to choose or customize the functionalities or change the design of their community's interface. With the increase of functionalities available for users in online communities, users should be allowed to customize their own interfaces to the community in order to improve attractiveness, convenience, decrease confusion due to unused functions, and increase their own performance in the community.

The survey of previous work on customization in user interfaces shows that two main approaches exist: adaptable and adaptive, with a compromise mixed-initiative approach to customization. When comparing the adaptable and adaptive approaches, the experiment with SMART MENUS by McGrenere et al. (2002) showed that 65% of the participants preferred adaptable interfaces. Also in the experiment done by Findlater and McGrenere (2004) for selection of functions from the menu, participants were faster in selection when they used adaptable interface. In this case, 55% of the participants preferred the adaptable version of the interface. In these two experiments participants were frustrated when the system did the customization (in case of adaptive approach). When comparing adaptable with mixed-initiative approach, experiments by Debevc et al., (1996) show that expert users (who have experience in

using the adaptive bar) preferred the adaptable approach which gives them full control, while novice users (who have no experience in using the adaptive bar) preferred the mixed-initiative approach.

From the discussion of these main customization approaches we can say that the preference for a particular approach is based on the system that is designed, how annoying or useful are the features that are present in the system, and the way the suggestions are made. So designing systems which are adaptive or mixed initiative and which are pleasing to the user is difficult. The systems with adaptable approaches allow users to have full control of their interface and are preferred by users over system-driven adaptations. Each user should be able to do customization at any time while performing a specific task. In the next chapter, I describe an approach to create a new CMS that does end-user customization.

CHAPTER 3
DESIGN AND IMPLEMENTATION OF A USER CUSTOMIZABLE COMMUNITY INTERFACE

To build a user customizable interface of an application there needs to be customization options provided by the application. To ensure that such options exist, I have developed a content management system (CMS), which allows creating online communities with customizable functionality and interface. Each user will be able to create their own community dashboard – their personal interface to one or more communities hosted by the CMS, by choosing the functionalities they want and by applying the customization options. Users can register for more than one community and can manage functionalities related to different communities on their community dashboard. MUCCD (Manju User Customizable Community Dashboard) CMS was developed by using Adobe's Flex and PHP. This CMS is different from the others in the sense that it allows users along with owners of the community to choose the functionalities and arrange them on their community dashboard.

3.1 Design of the Proposed CMS

The design of CMS contains two interfaces:

1) Full interface - containing a complete set of functionalities that are available in the application; and created by the community owner/designer.
2) Personal interface/dashboard - a subset of the full interface, containing functionalities selected and personalized in their appearance by the end-user.

The personal interface is an adaptable interface, containing the functionalities that the user needs and understands. In some cases the personal interface can have all the functionalities

which are present in the full interface but arranged specially according to the interests of users. Figure 3.1 shows the two interface design of my CMS.

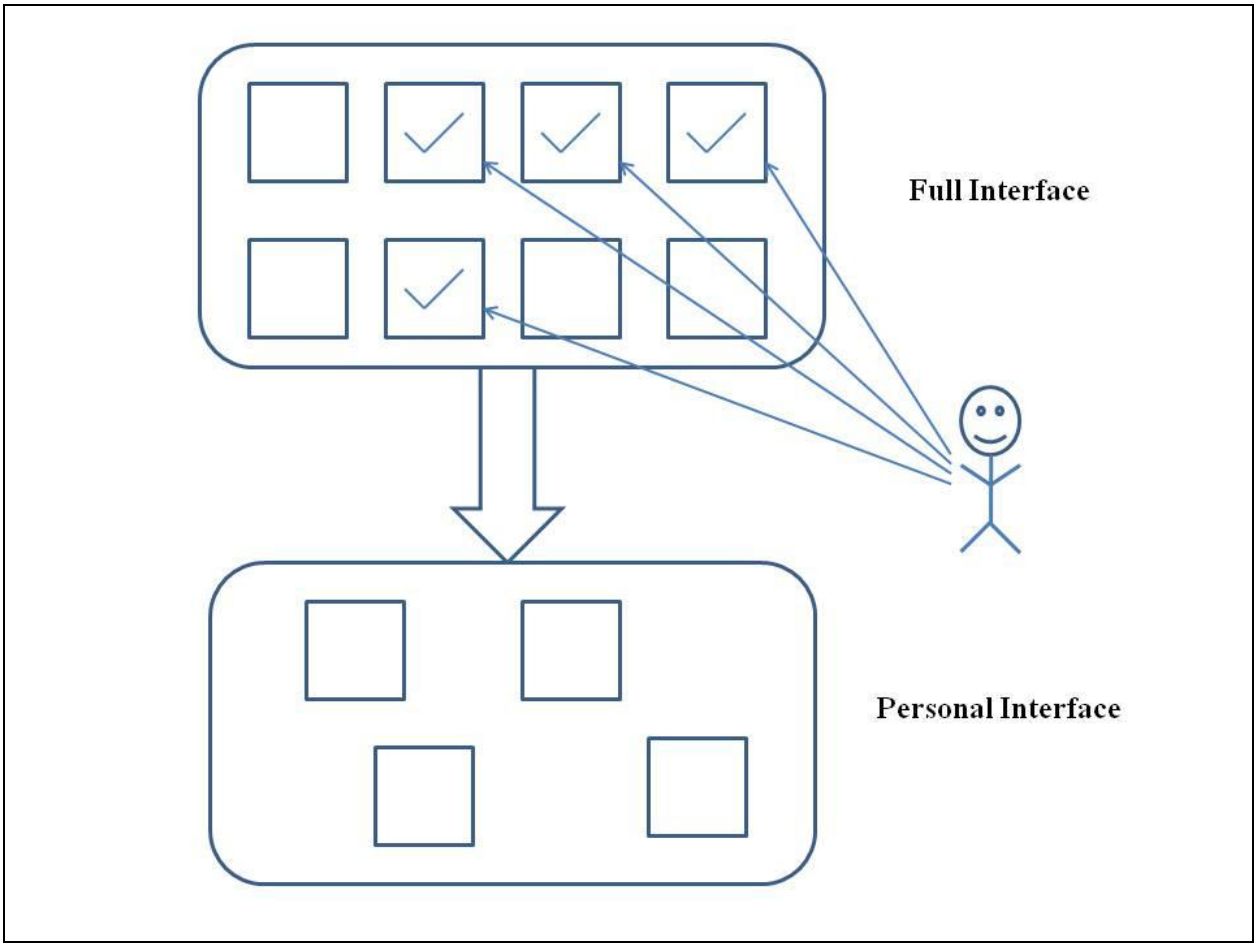


Figure 3.1: Two interfaces - Full and Personal

The design of the MUCCD CMS envisages two types of actors: the *community owner/developer*, and the *users*. The Owner/Developer is the person who will be developing the community using the proposed CMS. For example, in a school, this can be a member of the technical staff, or a teacher who is knowledgeable and willing to use the CMS to create online communities for his or her class(es). Users are the people who are involved with creating and

using the content in the community and managing the functionalities on their own personalized interfaces. These can be, for example, the students in a school.

The community developer/owner defines the community and creates the default interface. The default interface is created by choosing the functionalities which the community developer/owner thinks will be needed for his/her community and arranging the items representing these functionalities in the interface according to certain criteria or guidelines. The community is defined in terms of:

- What documents are stored, and what content is to be maintained, for example, shared files (text, PDF, video, audio), shared links/bookmarks, status updates, discussion entries, comments, ratings, etc.
- The ways of interaction with the community (how the users can communicate with others in the community). For example, submitting a file, submitting a link, chatting, commenting, replying, rating, etc.

The community developer/owner selects the functionalities that will be needed in the community from the full functionality set provided in the MUCCD CMS and defines the default interface for the users.

The users on the other hand have an option either to use the default interface which is created by the developer or create a personalized interface with the functionalities he/she needs. The users have an option to change the interface arrangement of all the functionalities according to their own preferences with the customization options that are provided.

The two-interface design for the MUCCD CMS contains the full interface, shown in Figure 3.2, and the personal interface, shown in Figure 3.3.

1. *Full Interface (FI):* Figure 3.2 shows the full interface of four communities hosted and developed in the CMS, as they appear to the user, where all the functionalities provided by the community designer(s)/owner(s) are shown. The four communities are focused on: travel, health, food and business. The owner of health community has chosen the functionalities related to sharing links, sharing files, searching for files or links, discussion forum for discussing about different topics related to health and the functionality related to the view the information of users registered for this community. From here users can select functionalities (by using a checkbox) which they think would be useful for them and they can add them to their personalized interface (community dashboard) by clicking the 'Add Items' button (circled in Figure 3.2). Users can also click the 'Use the default interface' button in order to have the full interface as their personal community dashboard.
2. *Personal Interface (PI):* Figure 3.3 shows the personal interface (dashboard) where the functionalities that are chosen by the user will be added. From the figure, we can see that the user is registered for travel and health communities and has added functionalities related to links, search and users accessing from both communities and files functionality from health community.

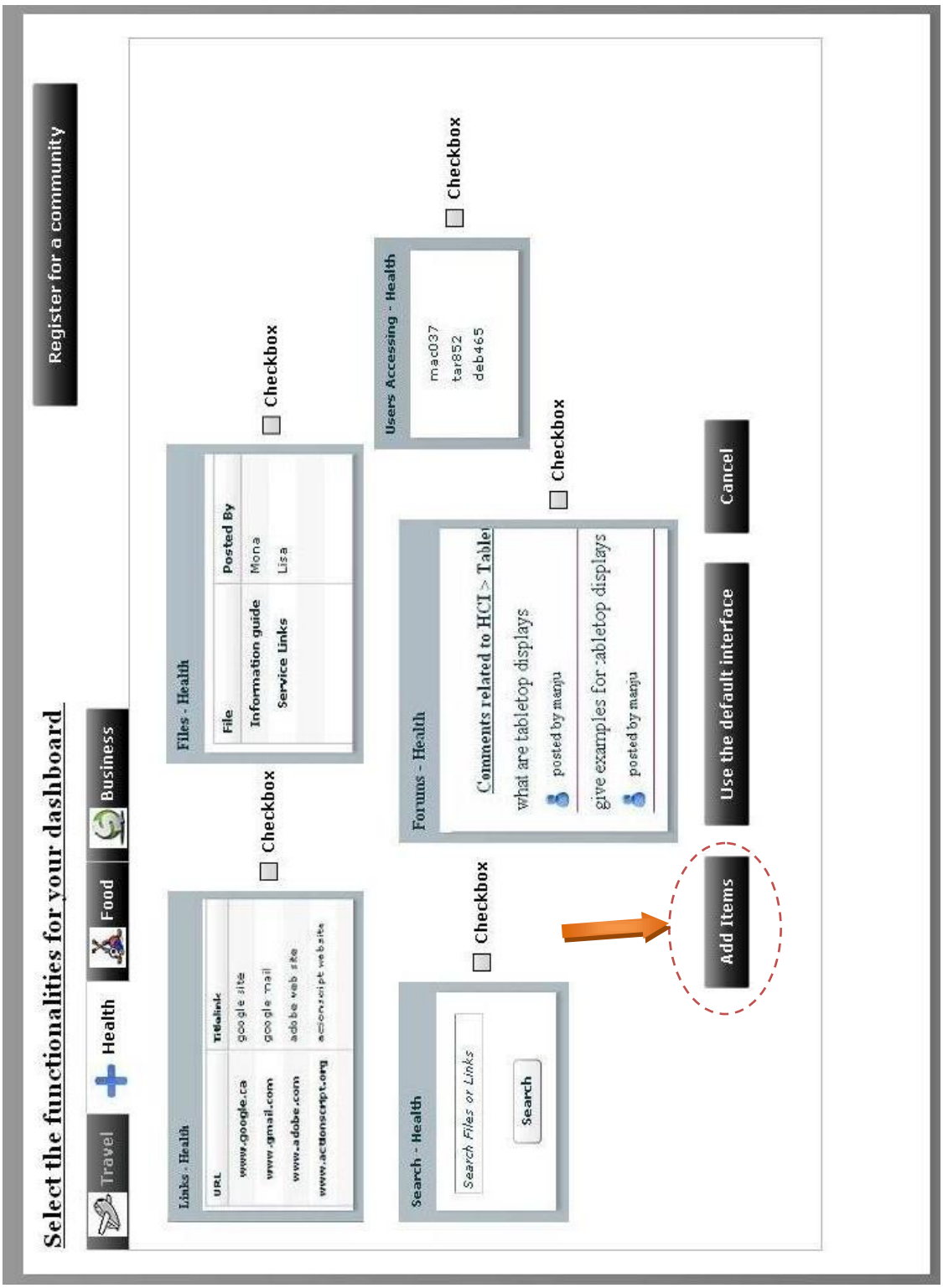


Figure 3.2: Full interface with the functionalities that are provided by the community owner

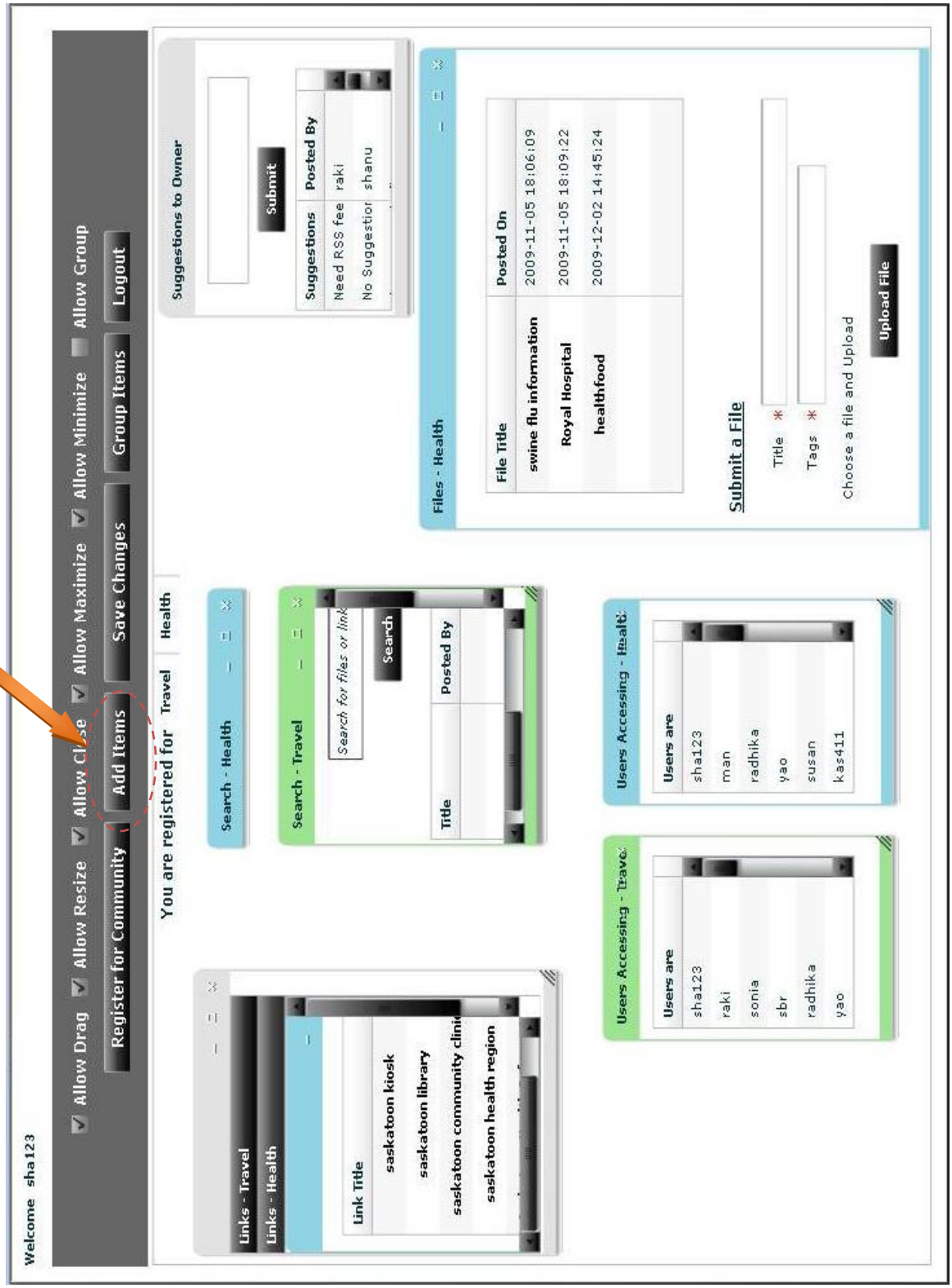


Figure 3.3: Personal Interface where the users can add the functionalities they want

With the two interface model, users can add the functionalities they want and they think would be needed in their community for their interaction with others. The users can always switch from their PI to the FI temporarily, in order to add new functionality boxes (using the button 'Add items' circled in Figure 3.3) as explained above.

Figure 3.4: Available customizable options for the user

The interface layout customization options that are available in the MUCCD CMS are shown in Figure 3.4. These options can be activated or deactivated by selecting the corresponding checkbox.

1) *Allow Drag:* By checking the allow drag checkbox, users will be able to move the boxes related to functionalities on the screen and place them anywhere on their community dashboard.
2) *Allow Resize:* By checking the allow resize checkbox, users can adjust the size of different functionalities. They can set the functionalities to any size to be viewed on the screen.
3) *Allow Close:* By checking this checkbox, users are given the ability to remove functionalities that are present on the screen. (If the users want to get back the functionalities that are removed from their personal interface, then they can go back to the full interface and add that particular functionality).

4) *Allow Maximize:* If there are lot of functionalities on the screen and their boxes are too small for the user to conveniently view the details related to some of the functionalities (e.g. the actual entries in a discussion forum), the user may want to resize the functionality boxes to whichever size they want. By checking the "allow maximize" checkbox, a "maximize" button will appear on the title bar of the functionality box, allowing the user to maximize that particular functionality whenever they want to use it. Once the user has maximized that particular functionality and wants to get back to the previous size the user can click the restore button, which appears in the place of maximize button.
5) *Allow Minimize:* When users don't want to see the entire box related to a particular functionality, then they can enable the minimize button by checking the "allow minimize" checkbox. By clicking the minimize button, only the title bar related to that particular functionality appears on the screen.
6) *Group Items:* If users want to group similar functionalities related to all communities or if they want to group the functionalities related to a particular community then they can do that by clicking the 'Group Items' button on the screen. By doing so, the user is provided with a box where he/she can drag and drop the functionalities which they want to group into that particular box. With this option, users can have the information related to similar functionalities or the information related to particular community at a single place so that it is easy to access.

In the MUCCD CMS users can use any of the customization strategies that are proposed by Bunt (2007) (discussed in section 2.4).Users can add functionalities to their personal interface at

any time while using the application: they can add the functionalities immediately after registering to the communities, or in the middle of using the application or they can opt for doing no customization by using the full interface.

3.2 CMS Functioning

In the MUCCD CMS users in the community perform GUI customization for their personal community (dashboard). Users are given an interface mechanism where they have full control of the interface appearance. Customization is based on the two interface model which was proposed by McGrenere et al., (2007). The owner of the community is provided with the full set of functionalities of the CMS. The community owner chooses the functionalities that might be needed for the community and arranges them on the community's homepage. The users who want to have access to that community need to register for that community and access the functionalities that are provided by the owner on the community's homepage. Now users can pick the functionalities that they need for their personalized community dashboard and add them to their personal interface and arrange them spatially as they wish. If users don't want to pick the functionalities or use all functionalities that are provided by owner of the community then they can use the default functionalities and interface that are provided by the community owner.

After choosing the functionalities, users can drag and drop the interface items (boxes) for these functionalities at any place on their community dashboard and can resize them to whichever size they want, group the boxes that provide access to functionalities related to a particular community or the similar functionalities related to different communities together.

In the proposed CMS any number of communities can be created and users should be registered for each community to access them. Users have an option to be registered for any number of communities. Once a user adds functionalities to the interface, he/she can arrange the functionality boxes (items) on the interface which can be done in different ways:

1) *Separating functionalities related to different communities*: If a user is registered for a single community, the position of the functionality boxes can be arranged according to the user's wishes. However, if user is registered for more than one community, she may want to group the functionalities related to different communities and arrange them spatially on the screen so that she distinguishes the functionalities related to each community, e.g. have a separate "space" for each community on her screen. The proposed CMS has a customization option to group the functionalities together. An example of this is shown in Figure 3.5 where the user is registered for two different communities (health and travel) and has placed functionalities related to different communities separately.

   Users can group any number and any type of functionalities at a single place. Users can group all functionalities related to single community so that they be accessed from a single box. An example of this is shown in Figure 3.6 where user has grouped the functionalities related to travel in one box and the functionalities related to health in another box.

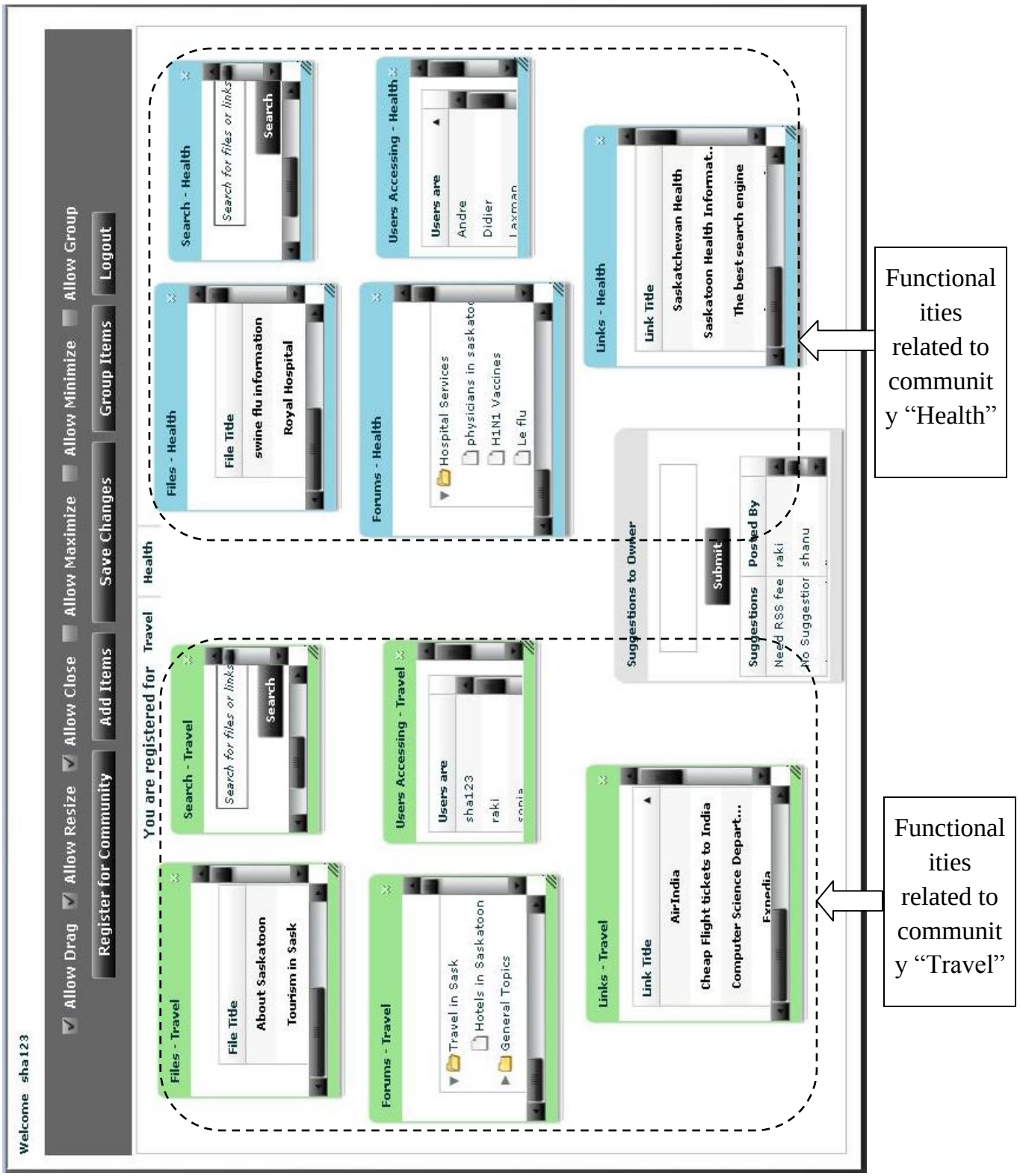


Figure 3.5: Separating functionalities related to different communities, Travel and Health

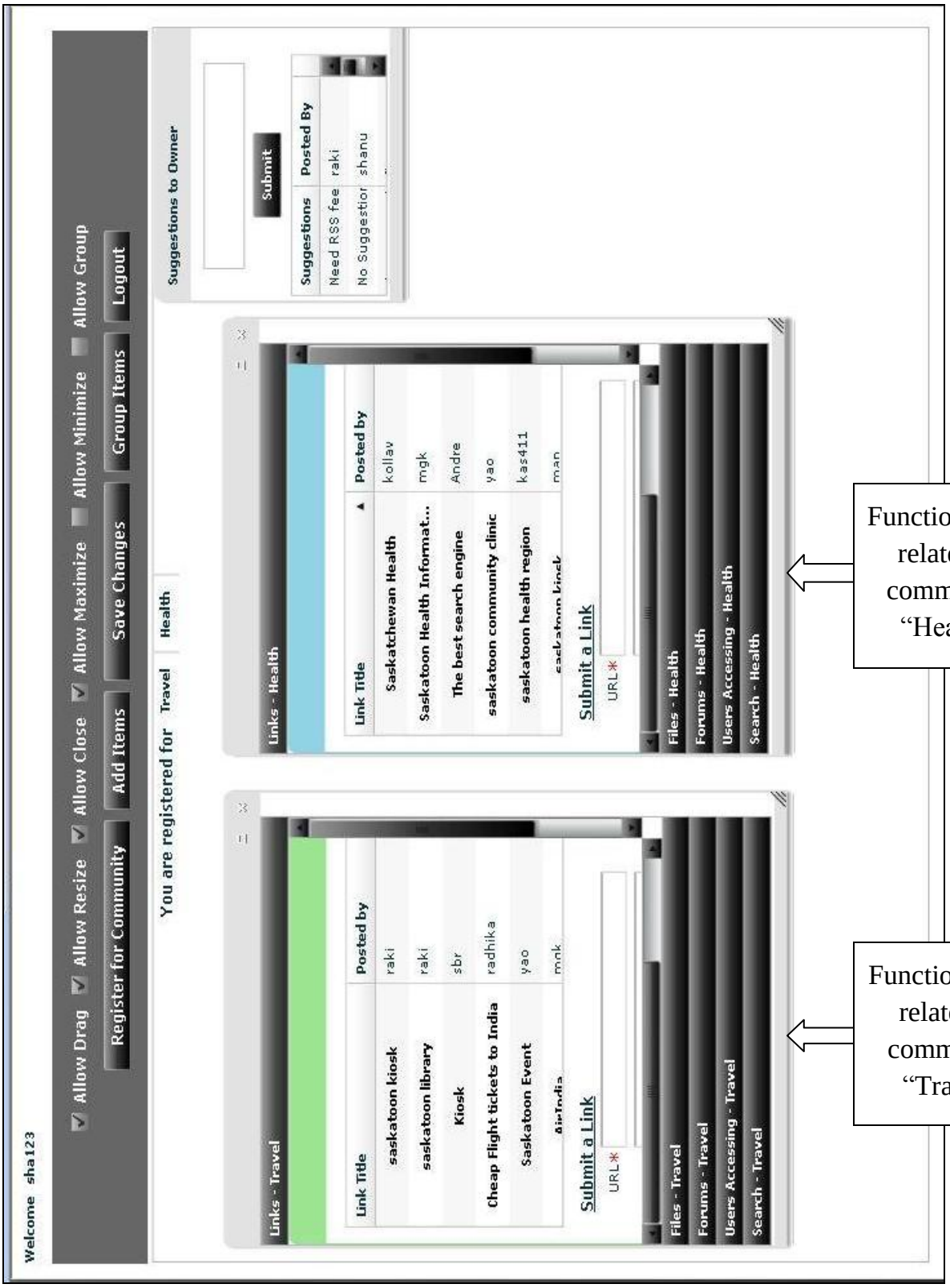


Figure 3.6: Grouping all functionalities related to single community

2) *Arranging similar functionalities related to different communities together*: If a user is registered for more than one community; then she can arrange the functionalities in such a way that similar functionalities related to different communities can be placed together. With this type of arrangement, users have an option to find similar functionalities at a single place and differentiation of functionalities from each other is also easier. An example of this can be seen Figure 3.7 where the user is registered for two communities and has placed similar functionalities from two communities adjacent to each other.

Users also have an option to group those similar functionalities related to different communities to have a single box representing a particular functionality. For example if a user has grouped discussion forum functionality related to two different communities, then it would be easier for that user to access all the discussion forums from a single box. An example of this is shown in Figure 3.8 where the user has grouped all the similar functionalities related to two different communities – health and travel.

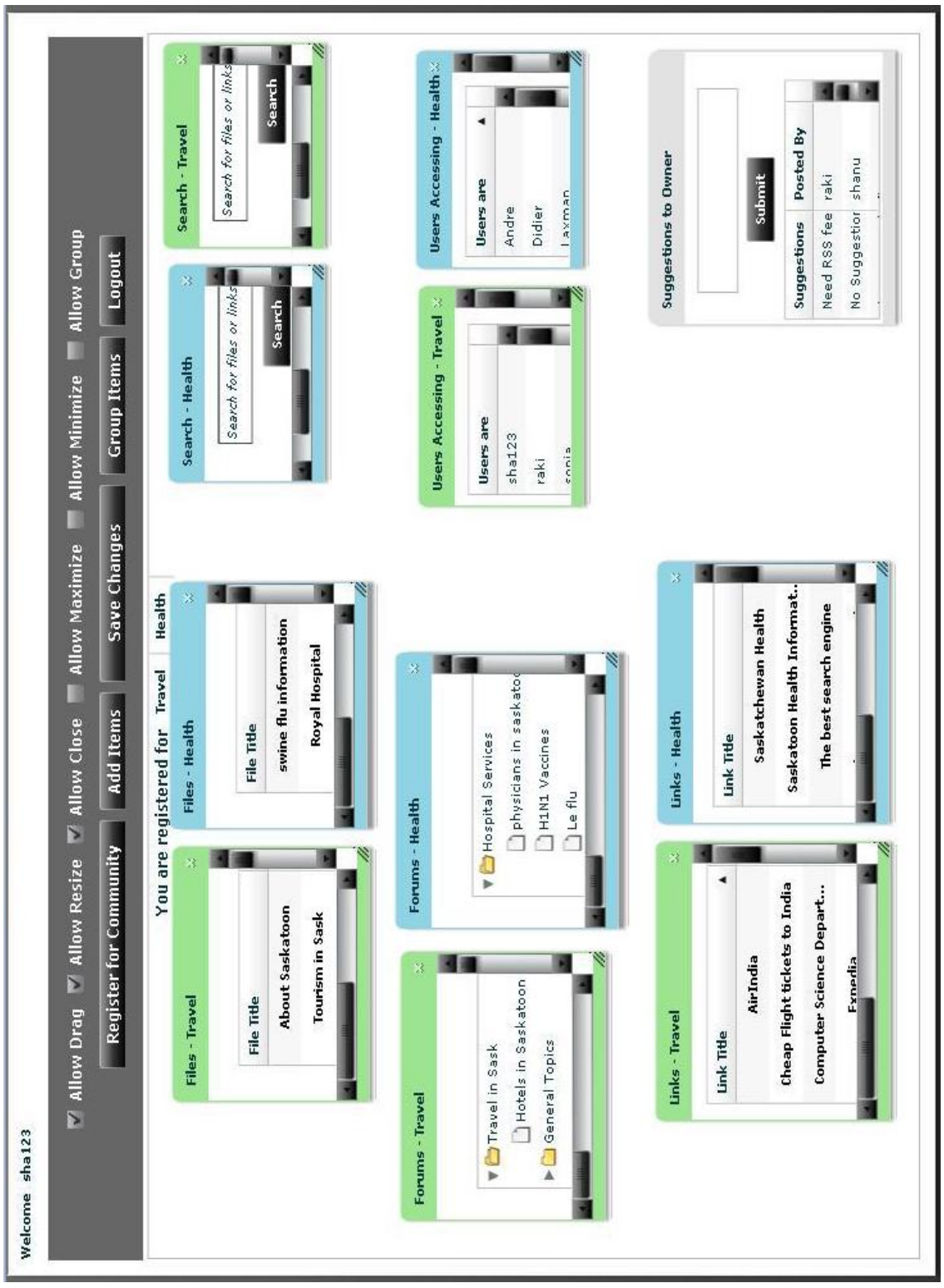


Figure 3.7: Arranging similar functionalities related to different communities adjacent to each other

Figure 3.8: Grouping similar functionalities related to different communities

3) *Grouping only a few functionalities*: Grouping functionalities can also be done by grouping only a few of the provided functionalities depending on the user's choice and leaving other functionalities separately. An example of this is shown in Figure 3.9 where the user has registered for two communities (travel and health) and has grouped only the functionalities related to forums and links.

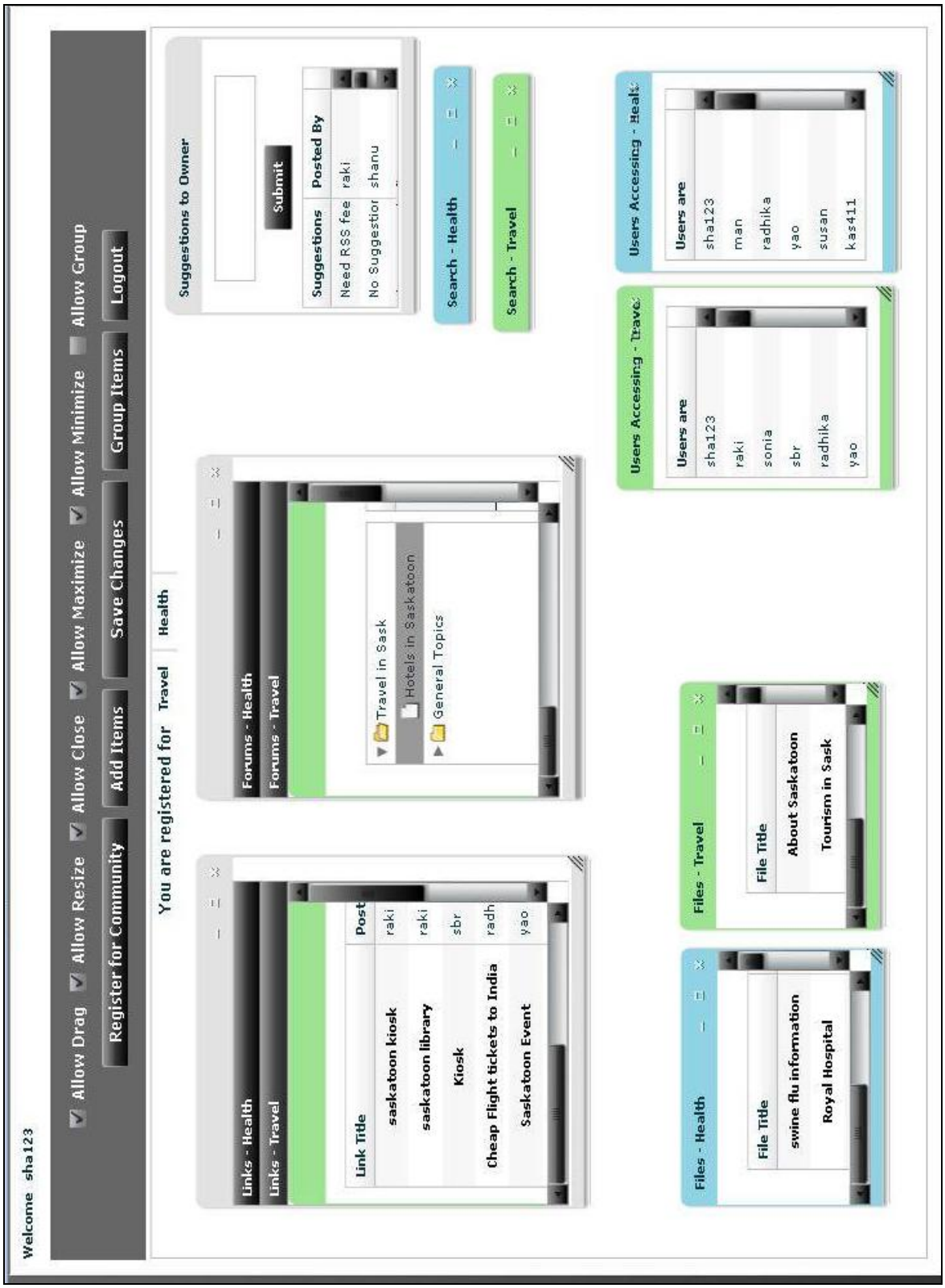


Figure 3.9: Grouping only few functionalities related to different communities together

3.3 Implementation of the proposed CMS

The MUCCD CMS is a web-based application, which uses Adobe's Flex, PHP and MySQL as implementation technologies. Flex was used to develop the client side of the application with the user interface. The output of a Flex application is in the form of animation (movie), which is downloaded to the browser at once, and avoids poor performance due to network delays, creating a near desktop quality of experience for the user. PHP was chosen as the server side language for developing this CMS. MySQL was used as the database for storing all the information related to the users and communities. The users first need to register into the application by giving a user id and password. Then they login into the application and register for the communities that they want to have access for. After registering to the communities they can see the functionalities that are offered by the community owner. The users can now add the functionalities useful or needed for them in their community dashboard. After adding the functionalities they can use all the customization options provided for them and place the functionalities as they like in which ever place they want.

3.3.1 Login

In order to use the CMS, each user should be registered with a user id and password. The application uses this id to store the functionalities and customizations related to the user and get the information back when the user logs in again. The user id is stored along with user content. The user can post links or files and can comment or post questions in the discussion forums. After registering users need to login into the application and select the communities which they want to have in their own community dashboard.

3.3.2 Technologies used

In earlier days to develop a web application which has some interactive features, HTML was used with the integration of scripting languages. Using recent technologies like java, .net with the database integration, more and more sophisticated applications are developed.

Each and every technology needs some skills and have some challenges associated with them while developing the application. This section presents the technologies used in developing this application in more detail.

PHP on the server

The CMS application was implemented using PHP (**P**HP: **H**ypertext **P**reprocessor) on the server side. PHP is a technology which provides server-side scripting, used to process the information that is obtained from the Flex client. PHP was created by Rasmus Lerdorf in 1995 (Wikipedia, PHP, 2009). It is free software released under PHP license. It can be deployed on most of the web servers like Apache, IIS and on different operating systems like Windows, Linux and UNIX. PHP allows the data from users to be stored in the form of session variable and allows users to have access to the information depending on his/her privileges. Different PHP files are written for each of the operation and the user's session is kept in track to decide which operation to perform. The data from PHP is stored in MySQL database.

Adobe Flex on the client

Adobe Flex is used in developing the CMS user interface (client). Flex is free open source software used to develop interactive web applications, which run on all major browsers, desktops and operating systems. For designing the application in Flex, MXML (Macromedia eXtensible Markup Language) is used, which is a declarative XML-based language, along with

ActionScript 3, a powerful object-oriented programming language. Flex has over 100 user interface components which can be used to build rich internet applications (RIAs). These RIAs can run on all major browsers with the help of Adobe's Flash Player and on desktops with the help of Adobe AIR (Adobe Integrated Runtime). Adobe's Flash Player and AIR are available for free download and the Flex applications which run on desktops using AIR are able to access the data that is present on the local machine (Flex Overview, 2009).

Adobe Flex Builder 3 was used for developing the application. Flex Builder 3 includes debuggers, compilers, component library, a visual interface for designing the user interface and it will enable users to do intelligent coding.

Flex was chosen because of the following features (Flex 3 benefits, 2009).

1) *Rich User Experience*: Applications can be developed for both web and desktop using Flex which provide high interactivity and which have very expressive interfaces. These applications can reach more users and they can be satisfied with these applications.
2) *Cross Platform Accessible*: Flex applications run using Adobe's Flash Player and this player is installed on over 98% of the internet connected computers. Flash Player is also accessible across different browsers and different platforms.
3) *Ease of Use*: Development of application is easy in Flex by using the application services, libraries, skins and containers. It also uses wizards to connect to the existing web services or for generating database connection code in PHP, Java, ASP.NET and Adobe ColdFusion.
4) *Open Source*: Flex 3 is open source software and it provides support for designing different patterns of web applications using the modern programming model.

5) *Scalability*: Using Flex we can develop scalable applications from simple components that can be included in the existing website to complex website or any desktop application. It has a prebuilt library for the components which allows users to create RIAs of all kinds.

Flex Builder 3 IDE is based on Eclipse and it includes editors for writing MXML, ActionScript and CSS (Cascade Style Sheet) code while differentiating the syntax coloring, code collapsing, statement completion and much more features.

MySQL

MySQL is the database used in my CMS to store the information related to users, customizations and functionalities they have used. MySQL which stands for My Structured Query Language is an open source relational database management system (RDBMS). MySQL works on different platforms and is written in C and C++. Some of the features of MySQL are:

1) *Web and data warehouse strengths*: The query engine has a very high performance with fast insertion capability and also supports fast full text searches.
2) *Strong data protection*: This database can be accessed only by authorized users and there is very powerful data encryption and decryption function.
3) *High availability*: It runs a very high-speed master/slave replication and there are specialized cluster servers for offering instant failover.
4) *High performance*: It provides table and index partitioning, distinctive memory caches, full text indexes, ultra fast load utilities and much more.
5) *Transactional support*: It has ACID (Atomic, Consistent, Isolated, Durable) transaction support, unlimited row-level locking, multi-version transaction support.

Architecture

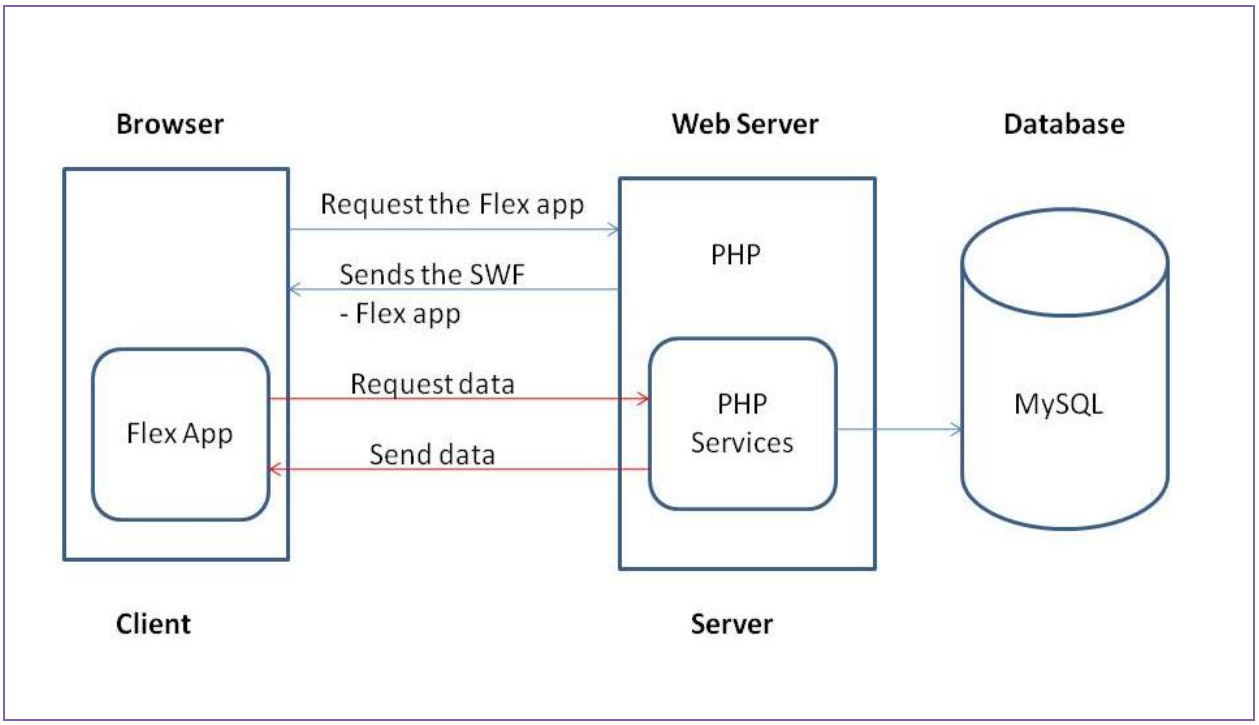


Figure 3.10: Flex and PHP architecture

Flex applications use SOA (Service Oriented Architecture) model, where Flex is used to create the client and the client connects to the data using services. While using Flex applications, when the browser makes a request, the server sends the complied Flex application in the form of a SWF file which runs inside the browser using a Flash Player plug-in. This SWF file holds only the client-side business logic. If data is needed from the database, then the Flex application makes a request for data to the PHP services that reside on the web server. These PHP services interact with the MySQL database to get the required data (shown in Figure 3.10). The server then sends only the data in the form of XML to the client and the client knows how to represent the data visually. This application can change state without refreshing the page or reloading the SWF file in the browser. The Flex client can be connected to the PHP back-end in two ways:

over HTTP and by using sockets. Over HTTP, there are four different ways to connect to the server: REST style services, web services (WSDL/SOAP), remoting (or RPC), and XML-RPC. I have used HTTPService class to use the REST style services. The request is sent from the client using the POST variables and the response is received in the form of XML, JSON or custom formatting (Corlan, 2010).

The data model for the CMS is shown in the Figure 3.11. It contains 15 tables in a single MySQL database. Boxes in the figure represent the tables, dotted lines represent the foreign key relationship with the tables and the solid line represents foreign key relationship for the primary key in a particular table. A detailed description of each table is presented in Table 3.1.

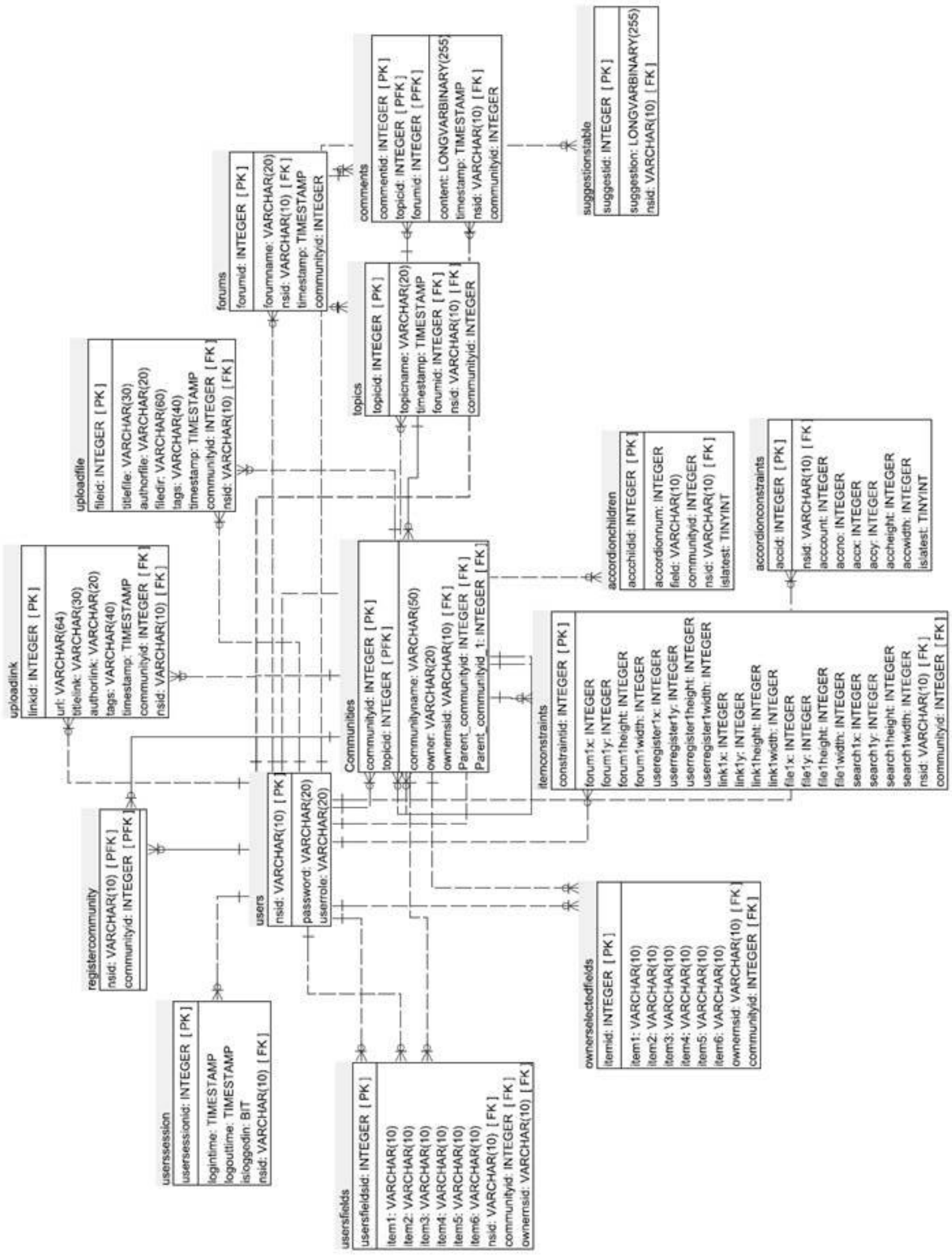


Figure 3.11: Data Model for the CMS

Table 3.1: Description of tables present in MySQL database

Table Name	Description
Users	To store the id, password and role of the user
Userssession	To store the session of the users after they login into the application
Communities	Description about the communities that are present in the CMS
Registercommunity	Stores the users and the communities that they are registered for
Ownerselectedfields	Stores the information of functionalities selected by the community owner for that particular community
Usersfields	Stores the information of functionalities that are selected by the users for each particular community
Itemconstraints	The constraints related to each functionality box - x and y coordinates, width and height are stored
Accordionchildren	Used to store the number of group boxes and the functionalities that are present in each group box for each user
Accordionconstraints	The constraints related to each group box - x and y coordinates, width and height are stored

Uploadlink	Used to store the links that are uploaded for each community
Uploadfile	Used to store the files that are uploaded for each community
Forums	Used to store the forums main topic name for each community
Topics	Used to store the forums sub topic name for each community
Comments	Used to store the comments for each topic based on the forum name for each community
Suggestiontable	Used to store the suggestions given by users for improving the community

3.3.3 Browser

To run the application a browser which supports the Adobe Flash Player can be used. The output of this application is an SWF (Shockwave Flash) file which runs only with the flash player support. With SWF file changing the state and the transition from one state to another would be difficult. For example, if a user wants to go back to a previous state or wants to undo changes, then the browser will not have a "back" button to load that particular page. The solution for this would be to use all the functionalities required to go back and forth in the application itself, which does not require the use of navigation buttons.

Figure 3.12 shows the block diagram of the proposed CMS, where the users need to login in order to access the communities. These communities have different functionalities which can

be accessed by the users. The logged in users can register for any number of communities and access either all the functionalities related to the communities or choose some of the functionalities that are provided by that community. Figure 3.12 shows three users registered for different communities (travel, health, food and business) and accessing the functionalities from those communities.

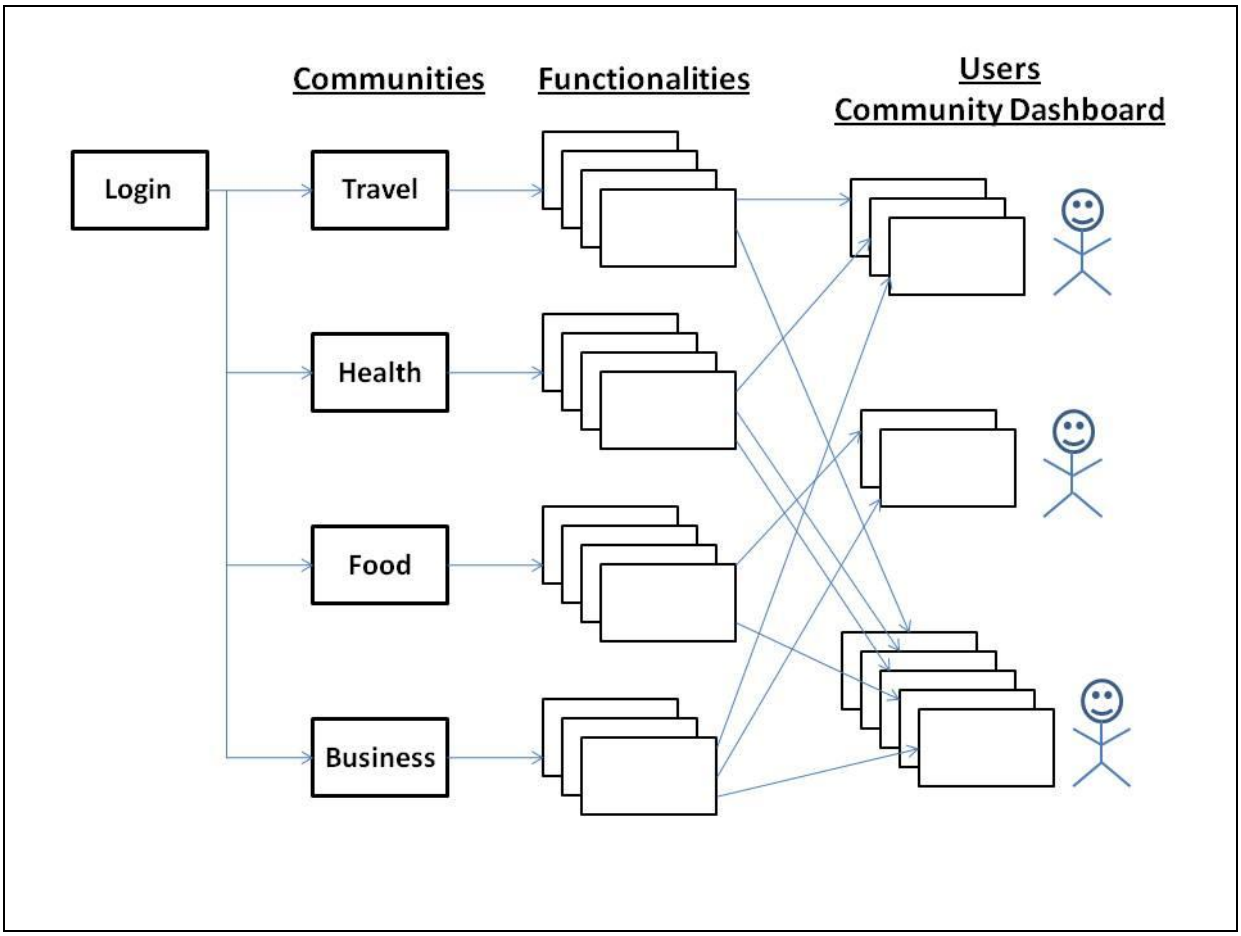


Figure 3.12: Block Diagram of CMS

3.4 Summary

The proposed CMS was developed using Adobe Flex with PHP which provides the users with access to two-interface model (full and personal interfaces). The information related to the users is stored in MySQL database. Users after registering to different communities can have access to all the functionalities that are provided by the owner of that particular community. Users can now arrange the functionalities that they have chosen on their community dashboard in a number of ways depending on their interest. They can also group the functionalities which they think needs to be at a single place. By using this CMS, users can create a personal interface/dashboard allowing them to access different communities and customize it to have different views for the functionalities related to each community depending on their preferences.

CHAPTER 4
EVALUATION

In the previous chapter, I have described the design and implementation of a CMS which allows users along with the owner of the community to customize the community interface and functionality selection. While the implementation provides a "proof of existence", i.e. that it is possible to create such a customizable CMS, the "proof of concept" requires to show that users would actually customize their community interfaces when given this option, and that the system provides a good support for the customization. Two studies (pilot and main study) were conducted to provide a proof of concept.

Originally a field-test was planned in the area of higher education. Higher education provides a natural application for the proposed LMS, since LMSs are traditionally used widely in education, and used to create different communities for different classes. Students can register in many classes at the same time, so that they are members of several different communities where they discuss and share materials related to the classes. Customizations could be used by both the teachers in designing the communities for their classes, and by the students who need a dashboard to view at one place the communities for all the classes they are taking simultaneously. However, the field test of such an application is beyond what can be achieved in the time-frame of a single M.Sc. thesis. A field study would require an application which would run for a long time, involving active communities and which is robust enough to handle load, deal with privacy and security issues etc. Therefore, I have chosen to evaluate my application on a small scale (with several new communities related to topics of common interest created specifically for the experiment), with a group of users who share and discuss materials in these communities. The users were provided with four online communities (the users in the role of

"owners" could create and add more communities to the CMS): Travel, Health, Food and Business. These communities which were seeded with some data were selected by me in advance. The evaluation tested the use of customization options by users who were designing their personal community dashboard and the usability of the GUI customization functions.

4.1 Hypothesis

The main questions that the evaluation aims to answer are:

- Is it a good idea to customize the functionalities?
- Do users actually use the customization options?
- Are users satisfied with the customization options that are provided in the community?
- Are users able to navigate between the personal interface and full interface?

The hypotheses related to these questions are:

- *Good Idea Hypothesis:* Users like the idea of choosing functionalities and customizing their personal community dashboard.
- *Usage Hypothesis:* The majority of the users will use the chance to choose the functionalities for their personal dashboard and will use the available customization options to arrange the interface of the dashboard according to their preferences.
- *Screen-size hypothesis:* In addition to the usage hypothesis, I made the conjecture that the size of the screen will affect the number of customizations made by users.

Smaller screens allow less space for clutter, so I expected to see the users do more customizations if they use the application in a smaller screen.

- *Satisfaction Hypothesis*: The majority of the users will be satisfied with the customization options that are provided.

- *Navigation Hypothesis:* Users will be able to switch easily between the personal dashboard interface and the full interface.

4.2 Experimental Conditions

Several choices had to be made about the design of the study, which affect the results that can be obtained related to the main questions of the evaluation. These choices are explained below:

- *Laboratory Study*: The study was conducted in a laboratory with users who were invited to participate by using the application for particular amount of time. The alternative choice would have been conducting a field study of real communities created by using the proposed CMS, and observation of the users as they customize their dashboard interface to these communities. As mentioned earlier, creating a successful community is a difficult task which takes a lot of time. If I failed to create several successful communities (e.g. not able to persuade instructors to use the LMS to create communities for their classes), the second approach would have been impossible. Therefore I decided to evaluate the system in a laboratory experiment.

- *Communities*: The previous decision led to the decision about the choice of communities. Instead of using authentic communities, the communities were specially created for the sake of running the experiment. These communities were pre-seeded with content to make them look more realistic. The participants have a choice of communities they can connect to the pre-defined ones. Real communities would have provided a context to observe authentic behavior of the users.

- *Functionalities*: There are five functionalities that are present in each community for sharing links, files, searching for files or links, discussion forums and the other to see the users who are registered in that particular community. These functionalities were chosen to allow users to contribute and share content among themselves and to interact with each other through forums. Participants have to choose the functionalities that are required for their community dashboard. The addition of functionalities to the user's community dashboard varies from one user to the other depending on the customizations done on their dashboard.

- *Screen Sizes*: Users were given an option to work on two different screen sizes. A bigger variety of screen sizes could have been used, including very small screens of mobile phones, but because of time constraints, only two conditions for screen size were tested: a typical laptop screen size and a net book screen size.

4.3 Methodology

Two user studies were carried out: a pilot study with 5 participants and a main study with 15 participants. They had to use the community dashboard to access the pre-defined and seeded

communities. The information related to each user's functionality selection and personal dashboard GUI customizations was automatically collected and stored in the database. The data contained the communities each user was registered for, what functionalities were added for each community, the position and size of functionalities, how many groups the user had created, what functionalities were grouped and the information related to the user's interaction with the functionalities. The customizations that were performed were observed by me and were recorded by a screen capturing video. Finally, each participant filled out a questionnaire to give their overall experience in using the application.

4.3.1 Procedure

Participants for the study were recruited though email notifications that were send to them personally. Initially participants of the study signed a consent form stating their agreement to participate in the study. The study, including the consent form for participants and the questionnaire were approved by the Behavioural Research Ethics Board at the University of Saskatchewan (provided in Appendix A).

In the consent form that was given to the users, they were told that they need to use the application where they can do customizations and perform the tasks that were given to them. But the main point to be noted here is that even though they were told that they can do the customization the users had the choice for not doing the customization and using the default interface that is provided by the community owner by clicking on the "use default interface" button that was provided in the full interface of the application. In using the customization options also the users had the choice of not using the customization options as the options were provided in the form of a checkboxes and not as default options for each functionality box.

First the participants filled a questionnaire about their demographic data. Then a 10 minute introduction to the application was given to the participants regarding the screens, customization options and different functionalities in the form of a PowerPoint presentation. The participants were also provided with the same information in the form of text on their login screens. Then after logging into the application, the participants were asked to register according to their interest into online communities that were provided for the experiment. Once registered for the communities, the participants were able to access the functionalities that were provided in that particular community by the community owner/designer (the *full dashboard interface*). They could choose the functionalities that they wanted, and by using the customization options (moving, resizing, dragging-dropping and grouping), they could arrange these functionalities on their *personal community dashboard* according to their preference. Once they were satisfied with the arrangement, they could interact with the other users in the respective communities by using the functionalities they had chosen in their personal dashboards. They were asked to use the communities to find information of interest and interact with the application. The participants were given a set of tasks to accomplish, for example posting links or files related to the community, or interacting with the other users in the discussion forum by answering a question or submitting a question. At the end, the participants filled out a second questionnaire related to the overall experience in using the application.

The application was tested on two different screens – a larger one (1550*920 pixels) and a smaller one (987*724 pixels) to see how users would arrange the functionalities when the screen size is restricted. These two screens sizes were assigned to two different versions of the application and were both presented on a computer with a screen resolution of 1600*1200 pixels. The participants first used the smaller screen to arrange functionalities and after 15 minutes

switched to the larger screen where they worked for another 15 minutes. We expected that the participants would do more customizations on a smaller screen so I thought of testing the application on a different screen sizes. For this I chose a typical laptop and a notepad screen size.

4.3.2 Tasks

Participants in the study were asked to perform some tasks after they are done customizing their community dashboard. They are asked to upload a file in the "files" functionality or upload a link in the "links" functionality related to a particular community or interact with other users in that community by answering a question or submitting a question in the "forums" functionality.

4.3.3 Evaluation tool

The tool that was used for the evaluation was a questionnaire. Two questionnaires were used. The first questionnaire aimed to collect demographic data about the participants (provided in Appendix B) and the second questionnaire aimed to collect information related to their experiences with the application.

The average time for completion of the study was 45 minutes which included the time for filling out the questionnaires (15 minutes). The first questionnaire consisted of 6 questions collecting general information about the participant's information related to computer skills, usage of web, how many social community sites they were registered in and their knowledge on the topics like Travel, Health, Food and Business (the topics of the predefined communities). The second questionnaire consisted of questions related to particular features of the application and the user satisfaction in using the customization options, the navigation in the application and the overall idea of using customization options for online communities.

4.4 Pilot Study

Before running the full experiment, a pilot study was conducted to test the experimental procedure, and to check if the questionnaire (provided in Appendix C) was unambiguous and obtained the needed information. Five volunteer participants were involved. Of these 4 participants were from computer science and the other in plant sciences. The participants were asked to use the application with the larger screen. The pilot study took approximately 30 minutes and was carried out at the MADMUC lab at the University of Saskatchewan. Based on the feedback, modifications were done to the questionnaire and some changes were made to the application. The instructions in the pilot study were given verbally and the participants did not understand the complete idea of the experiment. Based on the feedback obtained, in the main study the instructions were given in the form of a PowerPoint presentation which showed different screenshots of the CMS from which the participants could get a clear idea of the application. In the pilot study, all the functionality boxes related to different communities had the same color. This made it difficult for participants to identify different functionalities and differentiate between different communities. In the main study different colors were assigned for different communities so that the user could easily identify functionalities related to different communities.

4.5 Main Study

4.5.1 Participants

The participants were recruited from the Department of Computer Science at the University of Saskatchewan and from outside the University through personal email invitations. The number of participants on the study was 15 (12 from Computer Science and 3 from outside

the department). The participants' ages ranged from 18-39. They were paid $10 for taking part in a 45 minute experiment which took place in the MADMUC lab. Each participant was invited separately to work on both the screen sizes of the application. The users were not invited in a group to work all together at the same time. Initially the participants filled out a questionnaire collecting demographic data and data about their experience with and use of online communities. Participants were asked to sign a consent form approved by the University Behavioural Research Ethics Board. The participants were then able to use the dashboard application which gave them access to four different communities that were created for the purpose of the experiment and seeded with some initial posts collected from the web. Table 4.1 shows the information related to participants.

Table 4.1: Description of main study participants

Participant #	Gender	Age	#hrs of browsing internet/day	# of registered Communities	Background Computer Science
P1	M	18-29	1-5	0-2	Yes
P2	M	18-29	6-10	3-5	Yes
P3	F	18-29	>10	3-5	Yes
P4	F	30-39	1-5	0-2	Yes
P5	F	18-29	1-5	0-2	Yes
P6	M	30-39	>10	3-5	No
P7	F	30-39	>10	3-5	Yes
P8	M	18-29	1-5	3-5	Yes
P9	M	30-39	1-5	3-5	No
P10	M	30-39	1-5	3-5	Yes
P11	F	30-39	<1	0-2	No
P12	M	18-29	>10	0-2	Yes
P13	M	18-29	1-5	0-2	Yes
P14	M	18-29	1-5	3-5	Yes
P15	M	30-39	1-5	3-5	Yes

Demographics and choice of communities

A question was asked in the first questionnarie to gather the information related to the participants general knowledge of the given community topics (Travel, Health, Food and Business). Participants rated their knowledge on a scale of 1 (beginner) to 5 (expert). All the participants gave the rating below 5 for all the topics except one participant for business (participant 7) and other for food (participant 12) as 5. One participant gave the rating 2 and below 2 for all the topics (participant 3), one other 3 (participant 5) for all the topics and one 4 (participant 11). For a few participants the level of knowledge varied for different topics. The information related to this is shown in Figure 4.1.

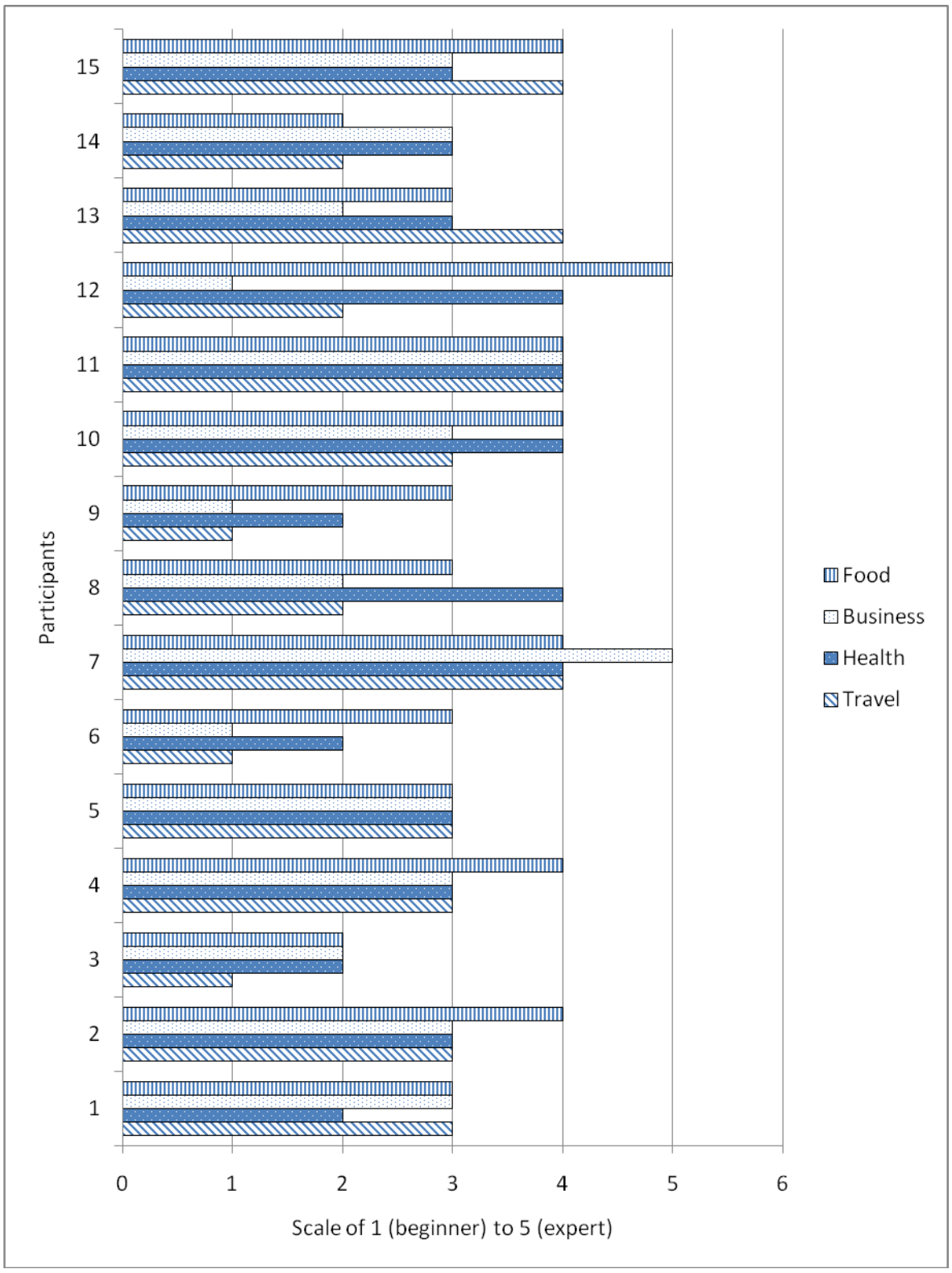


Figure 4.1: Participants rating their general knowledge on the given community topics

Figure 4.2 shows the percentage of users registered in the different communities that were provided in the CMS. Users were asked to register in at least three communities from the four communities that were available for them. All the users (15) registered in Travel community, 14 registered in Food, 13 registered in Health and 9 registered in Business. This shows that participants are most interested in Travel community.

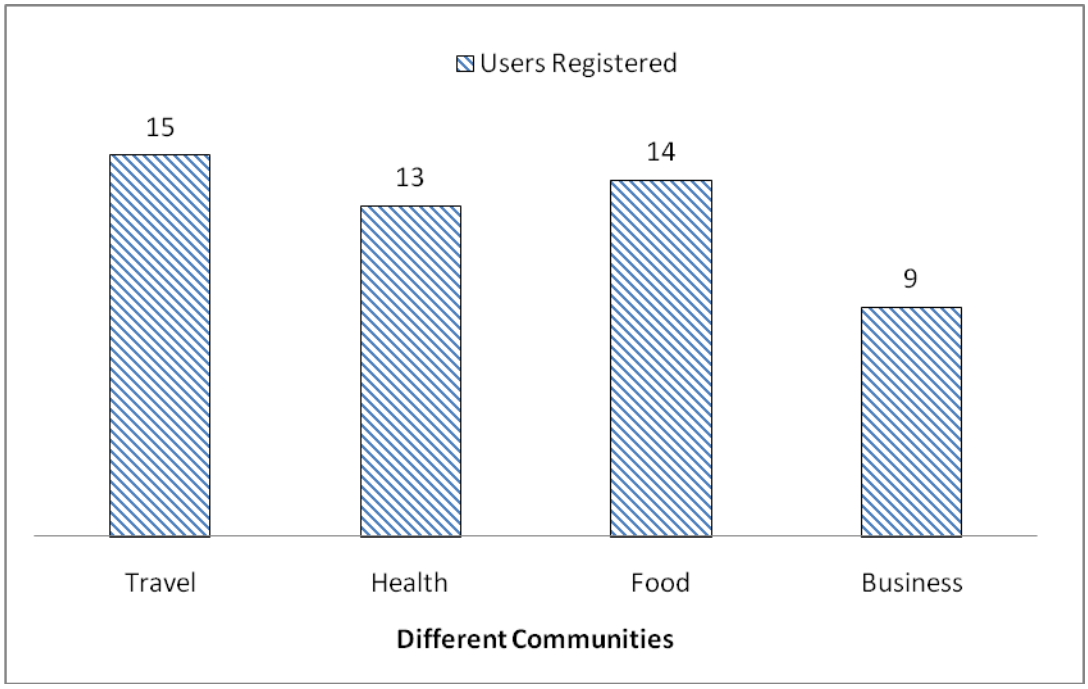


Figure 4.2: Users registered for different communities

4.5.2 Results

As explained in section 4.2.1, all the participants did the customization on both screens - large and small. The results in this section are related to the customizations done on both screens. Next, the questions that were asked in the second questionnaire are presented along with the results that were obtained.

The choice of functionalities across different communities

The results related to the participant's choice of functionalities across different communities are shown in Figure 4.3. As one can see, there are wide variations among the different forums in terms of functionalities commonly added by users. From the participants registered for business community, the majority (77.7%) added *"forums"* functionality (for discussion forum), while from the participants registered for travel community only half (53.3%) added this functionality. From the participants registered for business community, the majority (88.8%) added "*users accessing"* functionality (showing who are registered for a particular community), but from the participants registered for food community only 28.5% added this functionality. Again, from the participants registered for business community, the majority (77.7%) added *"links"* functionality (for sharing links on the topic of the community), but from the participants registered for the travel community only half (53.3%) added this functionality. From the participants registered for health community, the majority (69.2%) added *"search"* functionality, while half of the participants registered for food community added this functionality. The *"files"* functionality (for sharing files) was nearly equally popular among the participants registered for all communities, varying between 35.7% and 26.6% of the participants in these communities. Generally, the majority of participants in the business community added most of the functionalities, apart from "*files*", obviously seeing no value of sharing files in this community. The business community was selected by a smaller number of participants, but the percentage of the functionalities added by the users from that community was higher, so these participants were willing to use most of the functionality of this community. Some participants accessed the functionalities in such a way that they had similar functionalities from different communities and some others had different functionalities from different communities.

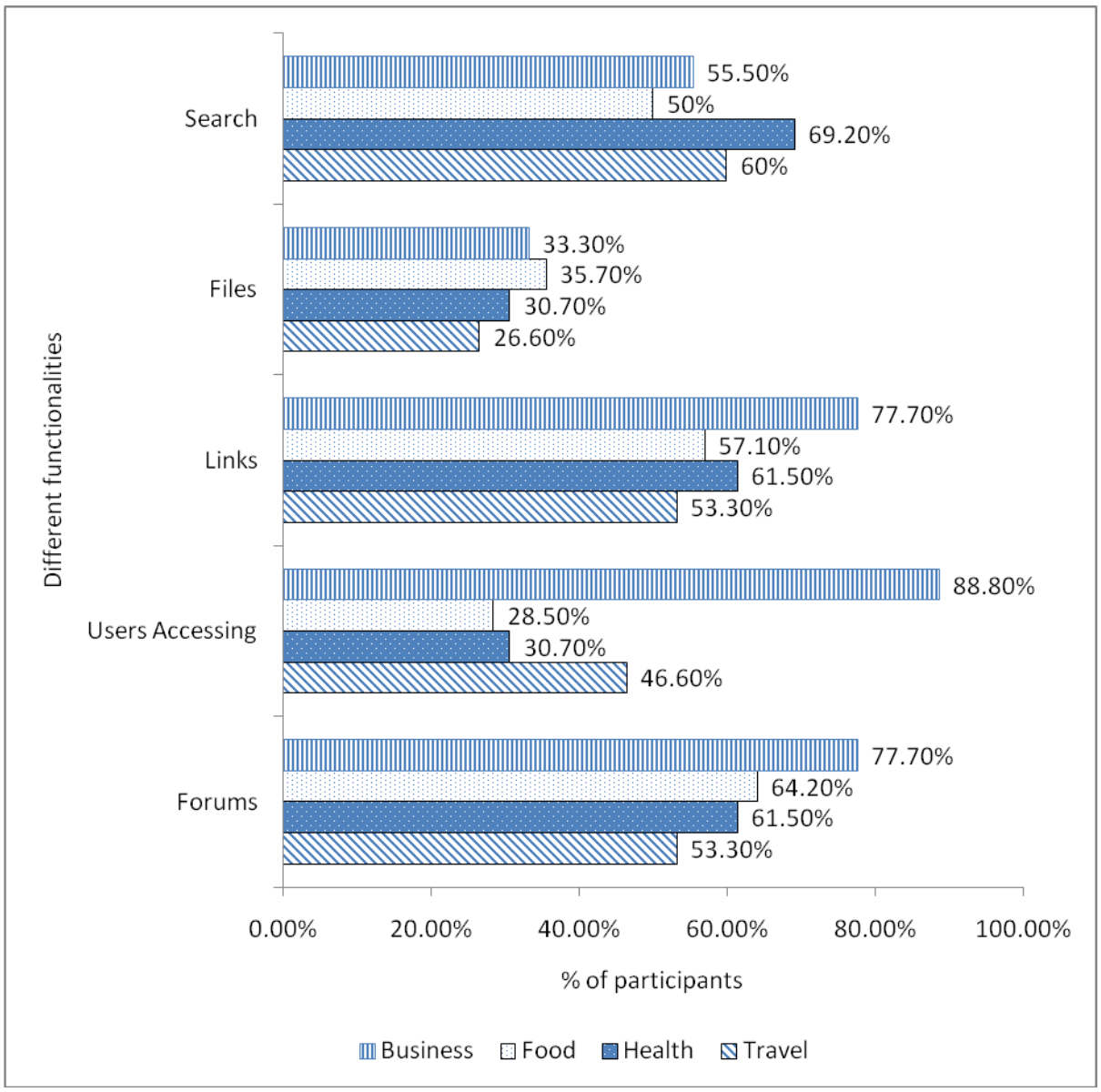


Figure 4.3: Percentage of functionalities added by participants from different communities

1) **Overall reaction to the application (1-Difficult to use and 10-Easy to use)**

1

This question was asked to know the overall reaction of users using MUCCD CMS. The participants were asked to rate the application in a range of 1 (Difficult to use) to 10 (Easy to use). 7% (one participant) chose the rating 10, 27% (four participants) chose the rating 9, 13% (two participants) chose the rating 8, 40% (six participants) chose the rating 7, 7% (one participant) chose the rating 6 and 6% (one participant) chose the rating 5.

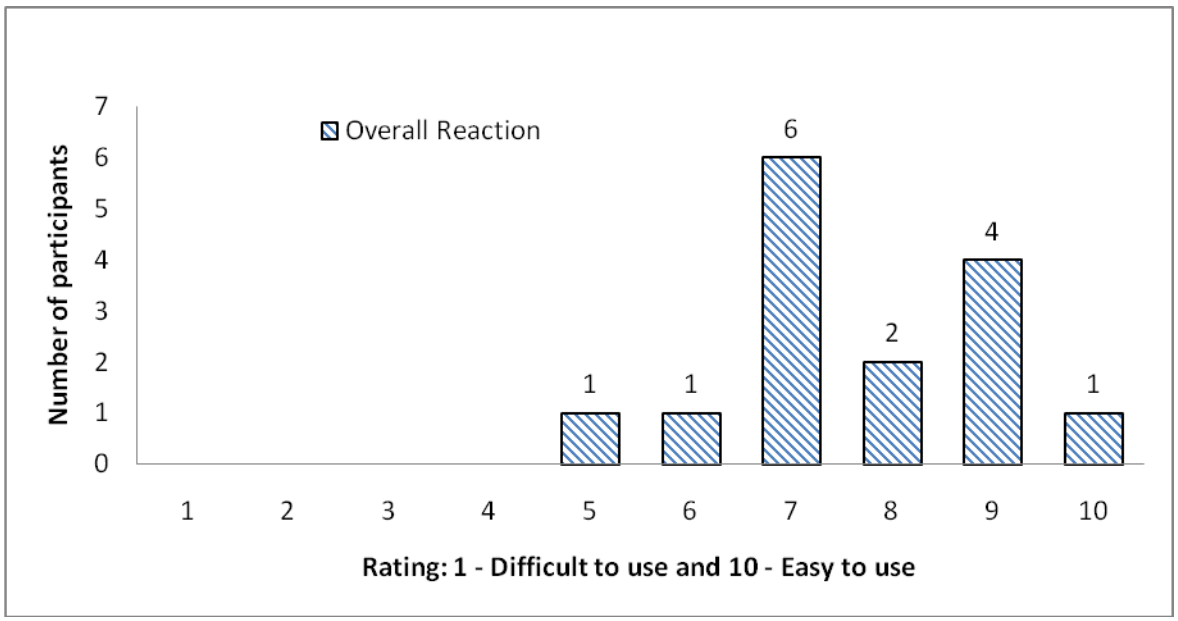


Figure 4.4: Participants reaction in using the application

2) **Do you think space is a hurdle for you in arranging the functionalities (or the boxes) in smaller interface?**

☐ YES ☐ NO

From Figure 4.5 we can see that 53% (eight participants) of the participants responded "Yes" and 47% (seven participants) "No". The participants who said "yes" where all with computer science background.

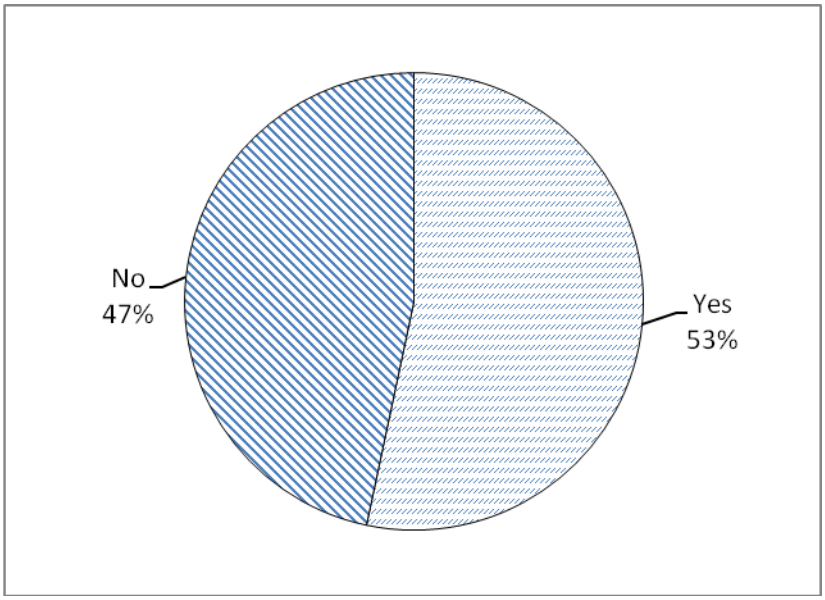


Figure 4.5: Space as hurdle in a smaller interface

3) **Did you use the customization options provided for you (like allow close, allow resize, allow maximize)?**

☐ YES ☐ NO

This question was asked to check if the participants ever used the customization options provided for them in arranging the functionalities on their community dashboard. 100% (all fifteen participants) said that they used the customization options because arranging functionalities cannot be done without using these options.

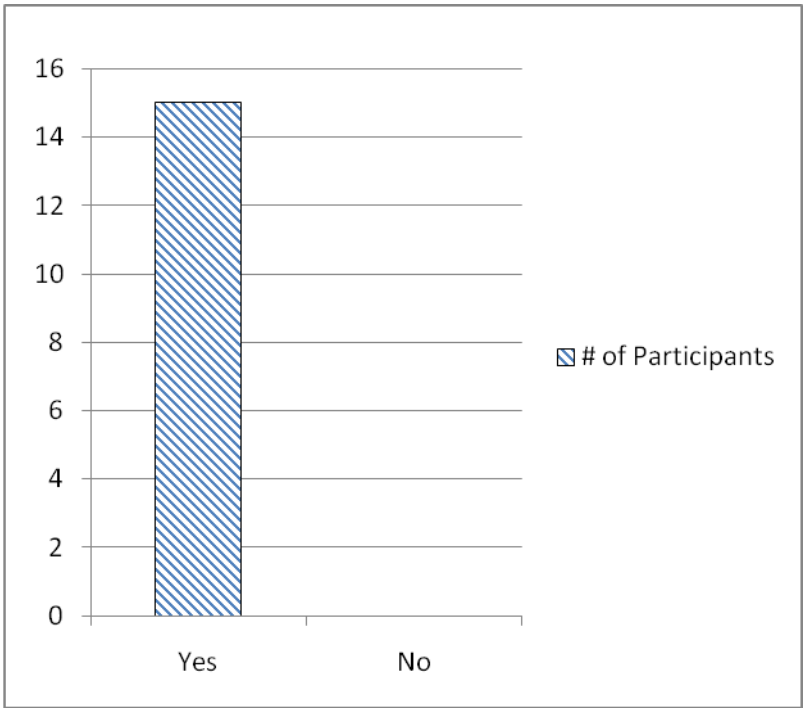


Figure 4.6: Use of customization Options

4) **Did you like the idea of choosing and arranging functionalities according to your interest?**

☐ YES ☐ NO

All fifteen participants answered "Yes", which supports the good idea hypothesis.

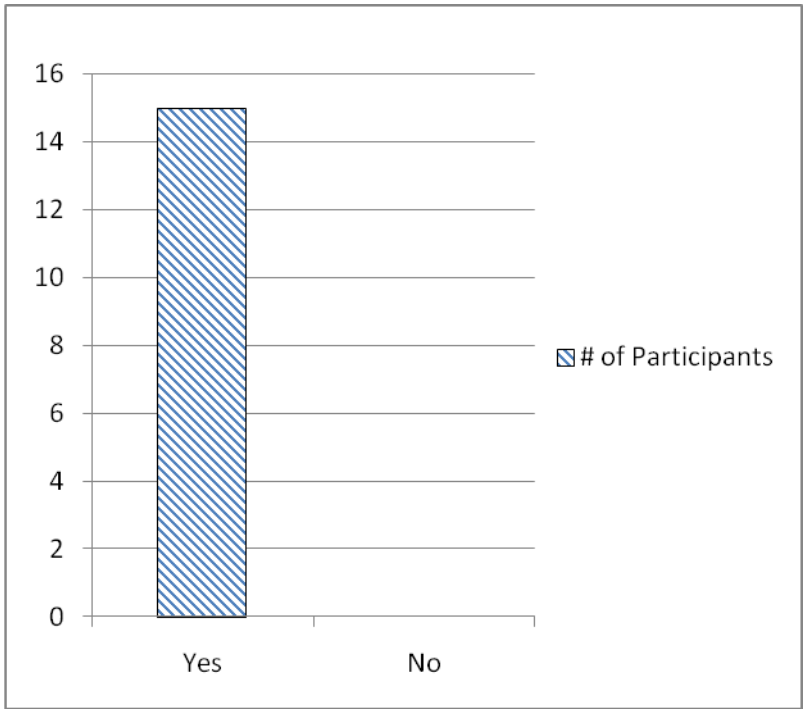


Figure 4.7: Like the idea of customization

5) Did colors help you in arranging the boxes?

☐ YES ☐ NO

Different colors were assigned to the boxes related to functionalities for different communities so that it would be easy for the users of the application to differentiate between the functionalities. Participants were asked whether the colors assigned helped them in arranging the functionalities according to their interest: 87% (thirteen participants) said "Yes" and 13% (two participants) said "No". The results show that colors were helpful for the majority of the participants in arranging the functionalities on their community's dashboard.

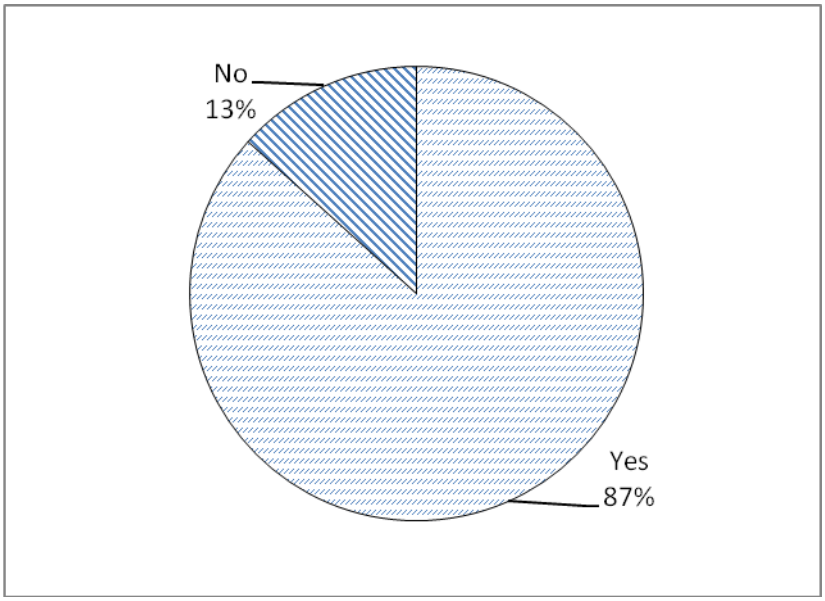


Figure 4.8: Use of colors in arranging functionalities

6) Were the following customization options useful for you in arranging the functionalities? (Please tick one box for each row)

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Closing	☐	☐	☐	☐	☐
Dragging & Dropping	☐	☐	☐	☐	☐
Grouping	☐	☐	☐	☐	☐
Maximizing	☐	☐	☐	☐	☐
Minimizing	☐	☐	☐	☐	☐
Moving	☐	☐	☐	☐	☐
Resizing	☐	☐	☐	☐	☐

The users of the application need to use the customization options in arranging functionalities on their community dashboard. The participants in the study were asked to indicate their level of agreement about the usefulness of customization options in arranging the functionalities. Most of them agreed that all the customization options were useful except minimizing and closing (see Figure 4.9). Minimizing was not that useful because the participants after adding the functionalities resized them to a particular size so that at least some part of the functionality was visible for them and did not minimize them (except for one participant who minimized the functionalities and placed them on his dashboard). Some participants minimized only to check if that particular option was working or not. After adding the functionalities to the dashboard most participants did not close them, so for some participants this option was not as useful as for others.

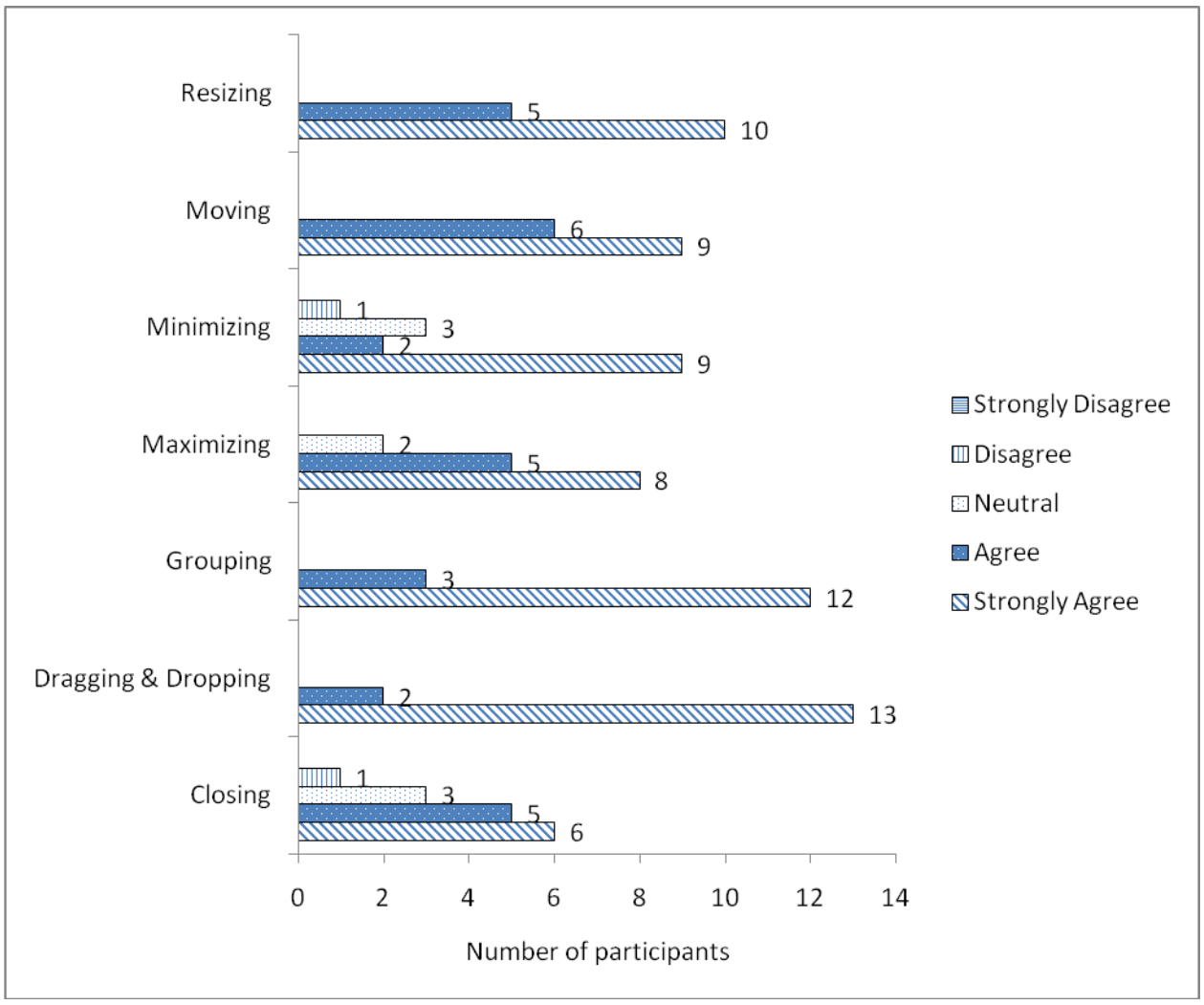


Figure 4.9: Usefulness of customization options in arranging functionalities

7) **Did you add all the functionalities that were available to your personal interface?**

☐ YES ☐ NO

Why? Please provide a comment:

Users of an online community might not need all the functionalities that were available for them, so in my application I have provided them the ability to choose the

functionalities which they need. This question was asked to find out whether the participants have added all the functionalities in the communities in which they are registered. 40% (six participants) said "Yes" and 60% (nine participants) said "No". This result shows that the majority of the participants did not resort to the default full interface with all the functionalities from their registered communities. Some participants did not want a particular functionality and some others thought that they might not use that functionality. During the observation, I noticed that one participant added all the functionalities related to one community and only a few functionalities from other communities. Interestingly, of the participants who said "Yes" all but one were with computer science background. Perhaps computer science students and graduates have a higher tolerance to interface clutter and "feature overload"?

Comments provided by the participants included:

a) Participant who answered "No" explained *"because I thought I would not contribute or make use of the ones I did not add, plus it gave me more space to work with".*
b) Participant who answered "No" said *"because at the time of testing I didn't think I needed to use them. But I think if I use the system in the future I might use all the functionalities".*
c) Participant who answered "Yes" said *"I would like to arrange all the items on my interface".*
d) Participant who answered "No" said *"I added the things which i needed in my personal interface".*
e) Participant who answered "Yes" said *"just try how they effect the interface [sic]".*

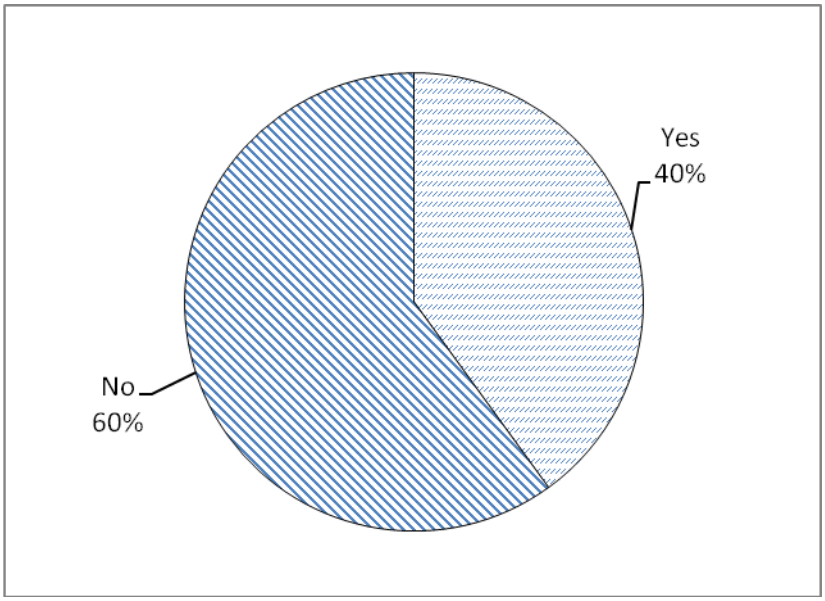


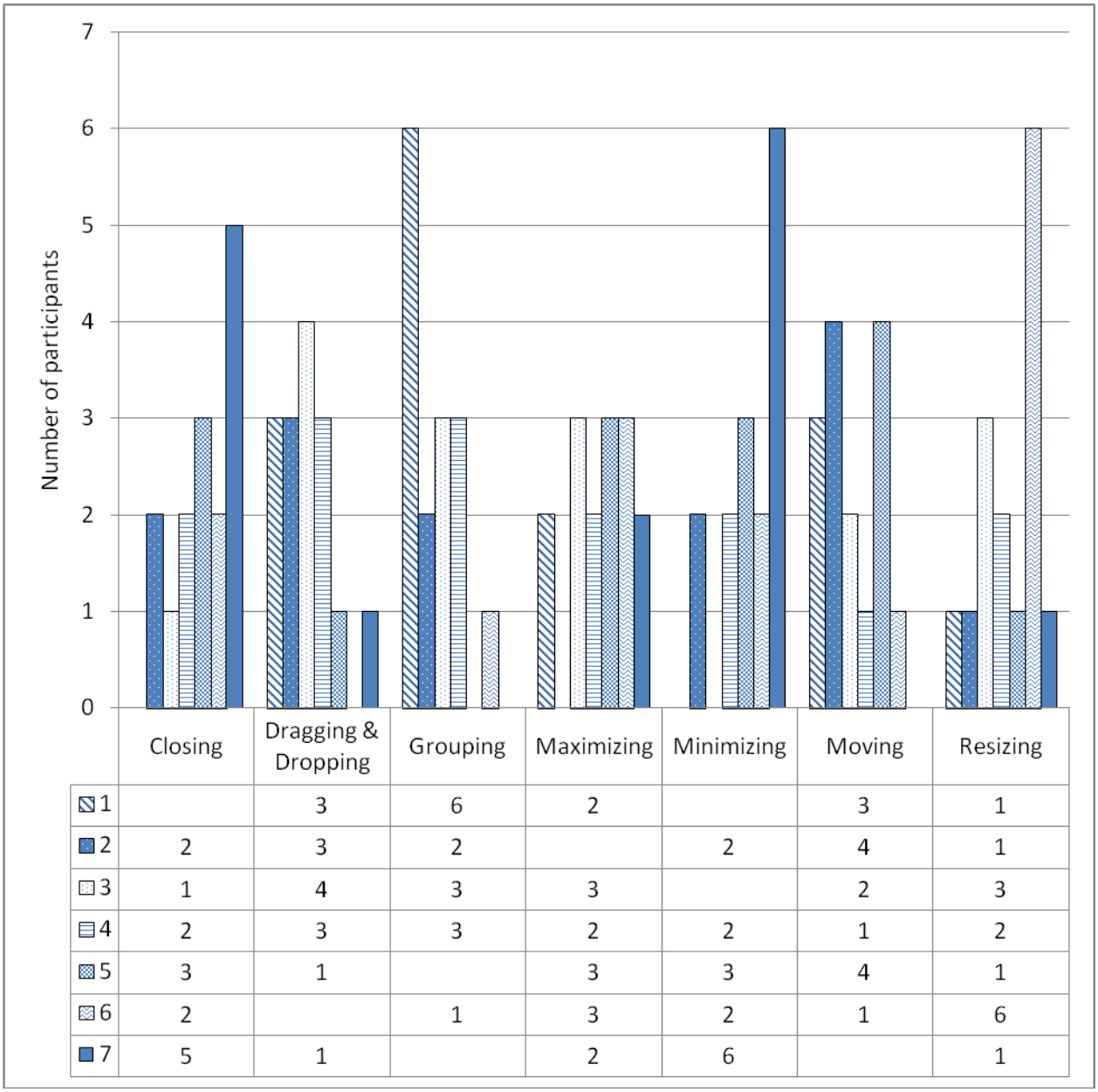
Figure 4.10: Adding all the available functionalities to the personal interface

8) **Please, rank these features from 1 to 7, according to their usefulness to you:** (1-most useful and 7 – least useful)

(Please do strict ranking. You can't have option to give same rank in two rows)

Closing	
Dragging &	
Grouping	
Maximizing	
Minimizing	
Moving	
Resizing	

For doing the customizations participants were provided with the customization options. I asked the participants to rank the customization options according to their usefulness in arranging the functionalities with 1 as most useful and 7 as least useful. From Figure 4.11 we can see that closing and minimizing were given the worst rank of 7 by 5 and 6 participants and the rest of the participants gave them ranks that were greater than 4. Dragging and dropping was given rankings less than 4 by almost all the participants except two who chose 5 and 7. Grouping was given rank 1 by 6 participants and the rest of the participants ranked it less than 4 except one who ranked it at 6. The maximization option got different ranks from 1 to 7 except for rank 2. Moving got a mixed result and was chosen as rank 2 by 4 participants and rank 5 by 4 participants. Resizing was given rank 6 by 6 participants; the rest of the participants gave it ranks that were spread throughout the scale. The results show that that closing, minimizing and resizing were not very useful for the participants. Dragging & dropping, grouping and moving were more useful for the participants. Maximizing got mixed results from the participants where the results were evenly distributed.



	Closing	Dragging & Dropping	Grouping	Maximizing	Minimizing	Moving	Resizing
1		3	6	2		3	1
2	2	3	2		2	4	1
3	1	4	3	3		2	3
4	2	3	3	2	2	1	2
5	3	1		3	3	4	1
6	2		1	3	2	1	6
7	5	1		2	6		1

Figure 4.11: Ranking the customization options

9) Do you think arranging all items takes lot of time?

☐ YES ☐ NOT SURE ☐ NO

Why? Please provide a comment:

After adding functionalities to the personal interface, participants need to arrange the functionalities on their community's dashboard according to their interest. So the participants in the experiment were asked if it took a lot of time for arranging functionalities after adding them to their personal interface: 27% (four participants) said "Yes", 73% (eleven participants) said "No". It shows that arrangement of functionalities does not take much time. All the participants who said that the arrangement of functionalities takes lot of time were with computer science background. Perhaps computer science students and graduates have less patience and want to get straight "to business" rather than to invest time in customization?

Comments provided by the participants included:

a) Participant who answered "No" said *"All customization requires some overhead, but if this is an application that I will be using regularly, it is ok to have time invested in the beginning to make the tasks that I perform easier later*"

b) Participant who answered "Yes" said "*After adding new items, one needs to do some more rearranging*".

c) Participant who answered "No" said "*It could take a lot of time if there are many items. With only five items per group, I think arranging items can be done quickly, especially after users are familiar with how the interface works*".

d) Participant who answered "No" said "*it depends on how we want to customize, I wanted my interface to be arranged according to my interest and I've added all the features to be arranged accordingly which took time*".

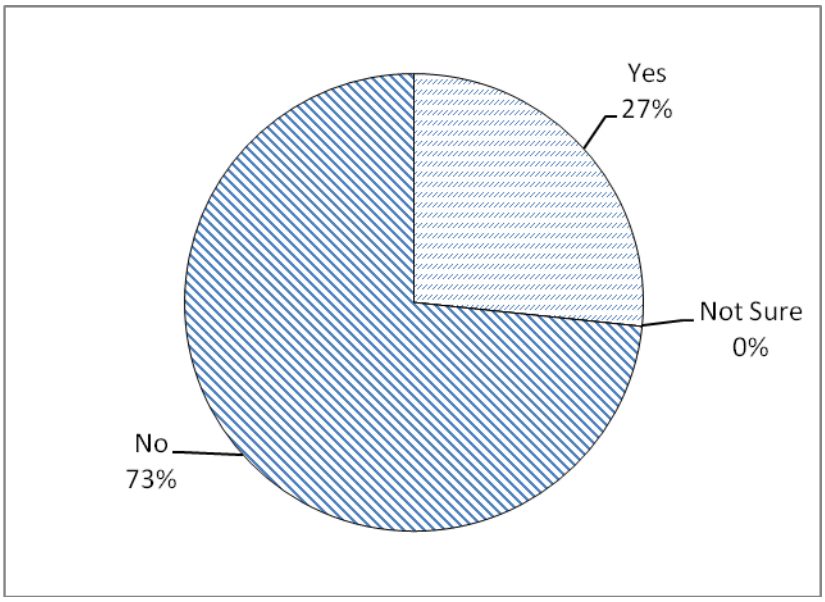


Figure 4.12: Does the customization of interface take lot of time?

10) **Did you like the idea of combining different communities at a single place? (By "different communities" I mean groups discussing different topics, e.g. health, travel, or different classes.)**

☐ YES ☐ NOT SURE ☐ NO

Why? Please provide a comment:

In my CMS, different communities are combined at a single place (the community dashboard) and users need to login only once to access those communities that are provided by the community owners. This question was asked to find out whether the participants liked the idea of combining different communities at a single place. Results showed that 87% (thirteen participants) said "Yes" and the other 13% (two participants)

said “Not Sure”. The results show that users like the idea of combining different communities at a single place.

Comments provided by the participants are:

a) Participant who answered “Yes” said “*One does not need to log into a different community to access a different group. Single log in is a good thing*”.

b) Participant who answered “Not Sure” said “*Combining different communities into a single place can help users get an overview of the communities. But, sometimes it's better to have separate places for different purposes*”.

c) Participant who answered “Yes” said “*because I can easily access and group common ideas and features from the different communities*”.

d) Participant who answered “Yes” said “*I like the idea of combining different communities so that it reduces the time for logging in and allows me check updates at a single location*".

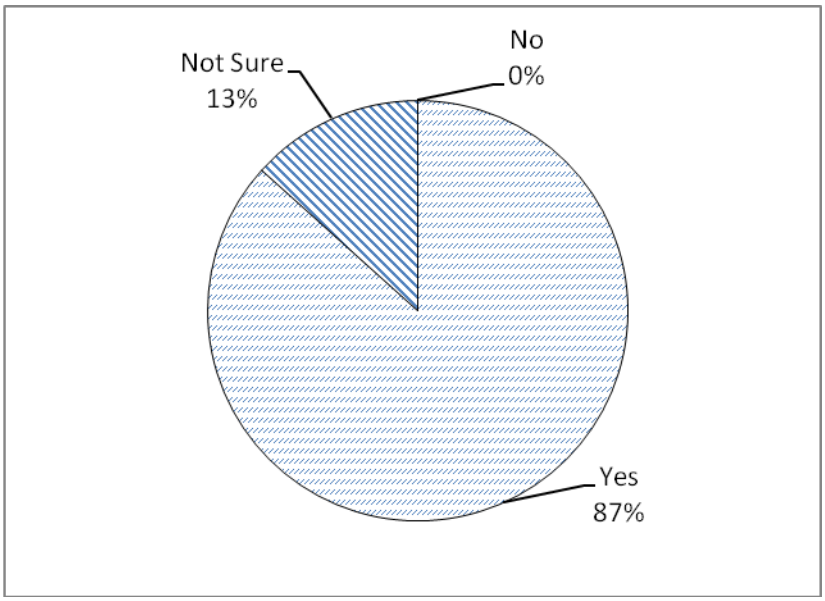


Figure 4.13: Do you like the idea of combining different communities

11) How difficult was it to do the following things.

	Very Difficult	Difficult	Easy	Very Easy
Identify the hidden functionalities* which are present in the communities homepage	☐	☐	☐	☐
Adding Communities	☐	☐	☐	☐
Add the functionalities using the checkboxes?	☐	☐	☐	☐
Using the customization options (e.g. allow close, allow resize, allow maximize etc) on the dashboard	☐	☐	☐	☐

*** Hidden functionality is the one that was not included in your dashboard (i.e. your main screen) or one which was closed by you.**

Participants need to navigate between their personal community interface and the full interface either for adding functionalities or for registering to new communities. All the participants agreed that it was easy to register for the communities which they were interested in. All participants except one said that it was easy to add functionalities related to different communities. Participants also found it easy to use the customization options that were provided for them. However, when asked how easy/difficult it was to identify hidden functionalities (i.e. the functionalities that were not added to the personal interface or the functionalities that were removed by the participant), five participants found it difficult. This might be because some participants had trouble understanding the meaning of the term "hidden functionalities" in the questionnaire even though they were provided with an explanation note at the end of the question. But it could also be due to

the difficulty of keeping in mind that the full interface of community always has all the functionalities, including those that were "customized away" by the user some time ago.

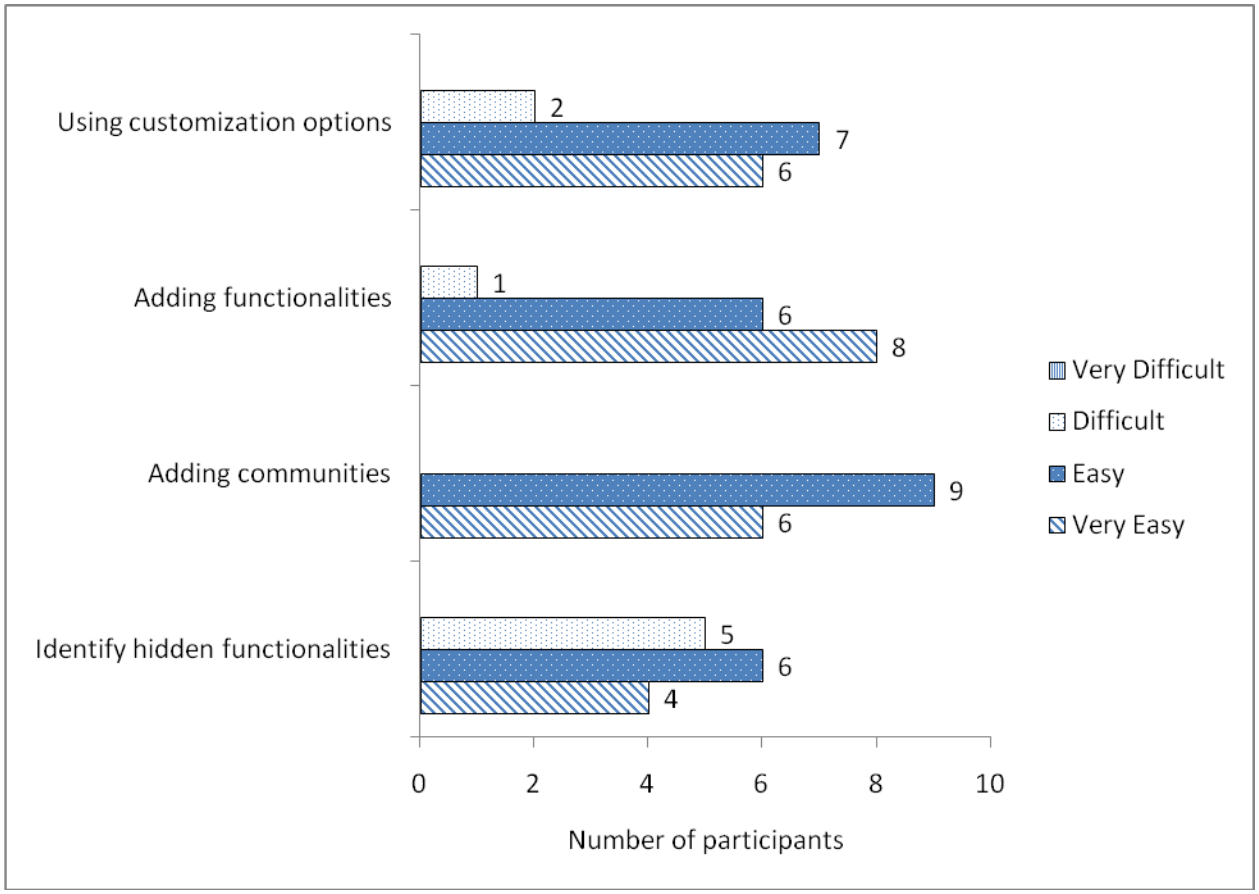


Figure 4.14: Navigation between interfaces

12) Were you able to notice the changes when selecting different customization options (like allow close, allow group)?

☐ YES ☐ NOT SURE ☐ NO

Users were provided with different customization options for arranging functionalities on their community dashboard. So when participants were asked if they identified these

changes when the customization options were selected, 93% (fourteen participants) said "Yes" and 7% (one participant) said "Not Sure". This shows that participants were able to identify the changes when the checkboxes related to the customization options were selected.

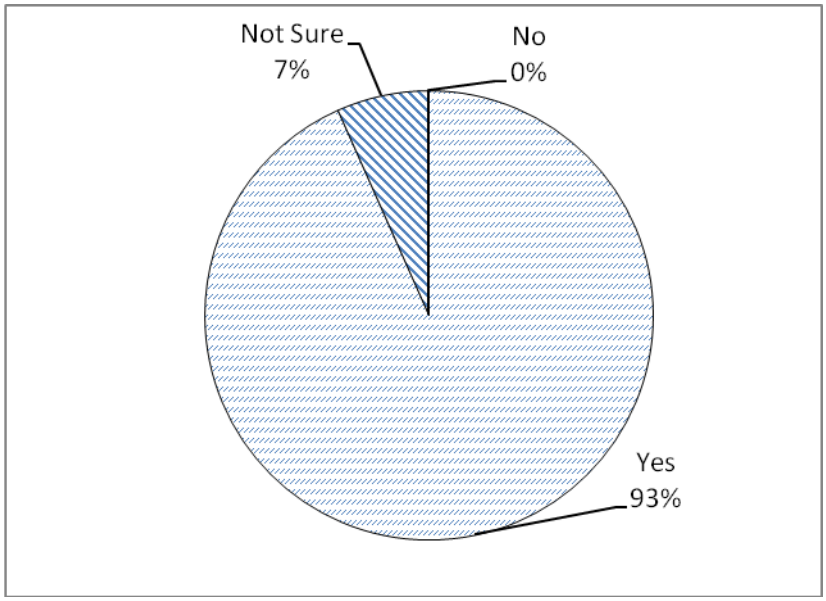


Figure 4.15: Identify changes when selecting customization options

13) Do you need any suggestions or tips interrupting you and telling to do a particular type of customization which might be useful for you?

☐ YES ☐ NOT SURE ☐ NO

Why? Please provide a comment:

My CMS is purely adaptable where users need to do the customizations and the system does not provide any suggestions or tips for doing customizations. So the participants were asked whether they need any suggestions or tips interrupting them and asking them

to do a particular type of customization that could be useful for them: 40% (six participants) said "Yes" and 60% (nine participants) said "No". However, the results from the observations and comments show that most of the participants need suggestions when they are using the application for the first time. Along with 3 participants who had computer science background, all the 3 participants with non-computer science background requested suggestions or tips while using the application. Of the users who asked for suggestions, 5 users are registered in 3-5 social networking sites and others were registered in less than 3 sites.

Below are some of the comments provided by the participants:

a) Participant who answered "Yes" said "*Good for the first time, but not again and again*".

b) Participant who answered "No" said "*After more time thinking and practice, I think it would be easy to figure that out*".

c) Participant who answered "No" said "*I don't think that such suggestions would be necessary to help me customize the user interface*".

d) Participant who answered "Yes" said "*saves lot of time and it would be more user-friendly*".

e) Participant who answered "No" said "*automated suggestion though useful sometimes are more often annoying*".

f) Participant who answered "No" said "*we get new ideas if we are given suggestions*".

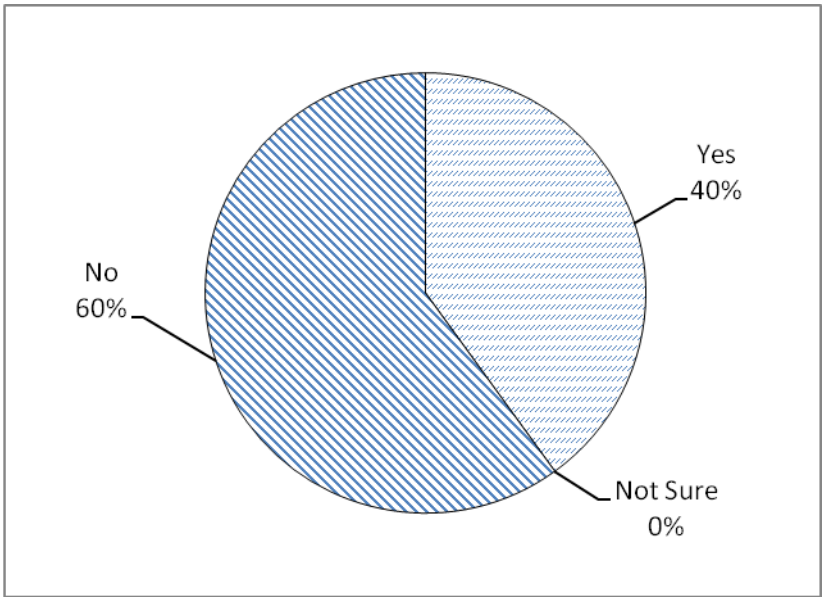


Figure 4.16: Need suggestions or tips for arrangement

14) Please indicate the extent to which you agree or disagree with the following statements:

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I customized to reduce the number of functionalities that I had in the full interface.	☐	☐	☐	☐	☐
I customized to make my personal interface as small as possible.	☐	☐	☐	☐	☐
I customized because I thought it would help me complete my tasks more quickly.	☐	☐	☐	☐	☐

The participants were asked about why they customized their interfaces. Participants were asked to indicate the level of agreement with 3 predefined options. 1) to complete their tasks quickly, 2) to make their own personal interface as small as possible, and 3) to reduce the number of functionalities that are present in the community's homepage. Figure 4.17 shows the summary of the responses. All fifteen participants (100%) agreed that they customized the interface to complete the given tasks quickly. Seven participants (46%) agreed that they customized to make the personal interface as small as possible, while four participants (26%) disagreed and the remaining four were neutral related to this reason. Nine participants (60%) agreed that they customized to reduce the number of functionalities, two participants (13%) disagreed with this, while the remaining four participants (26%) were neutral. From the results I can say that the participants mostly did the customizations in order to complete the given tasks quickly and these tasks were independent of each other. Some of them did the customization to reduce the number of functionalities that are present in their full interface. Finally few of them did customization to make their personal interface as small as possible.

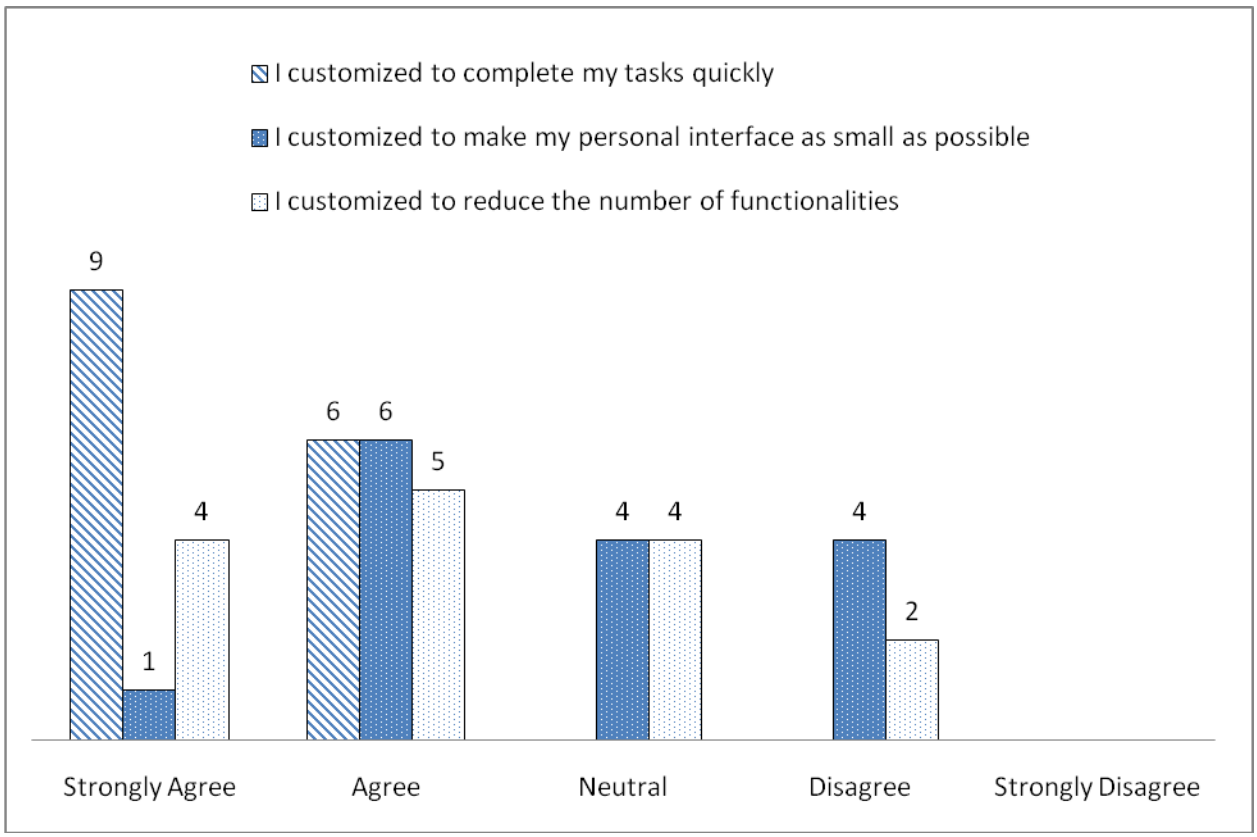


Figure 4.17: Reasons for customization

15) **Can you think of any other customization options that can be added to the application which might be useful?**

Why? Please provide a comment:

In my CMS I have covered the major customization options that can be used by the user to do the customization for their community dashboards. I also wanted to know other options that can be added to the CMS to make it more useful and interesting for the users to use the application. The following are some of the comments that are provided by the participants of the study. Participants are mostly satisfied with the customization options that are provided but also have their own suggestions for the application.

Comments provided by the participants are:

a) *Change colours*

b) *I think it covered the major options any additional options could make the system harder to use*.

c) *Having a default arrangement with proper window size would be helpful.*

d) *It would be good to choose colors for different communities by the users and also to use the same for grouping*.

e) *Choice of background colors, option of sizing the small boxes, Main search engines display by default. Precisely, anything whatever saves the user time in different categories with multiple levels in each*.

f) *how copying between groupings*

g) *it should give me the list of closed items on the side so that by clicking that I should get back all the items if I need them again*

h) *reset button would be good and useful*

i) *divide functionilities [sic] into tabs*

16) Generally, what do you think of the idea of choosing, adding and configuring your personal interface of the community?

☐ Very Good ☐ Good ☐ Bad ☐ Very Bad

When asked about what participants think of the idea of choosing, adding and configuring their personal interface: 60% (nine participants) said "Very Good" and 40% (six participants) said "Good". The result infers that all the participants think the idea of

choosing and adding the functionalities that are needed for their personal interface and configuring their community dashboard is very good.

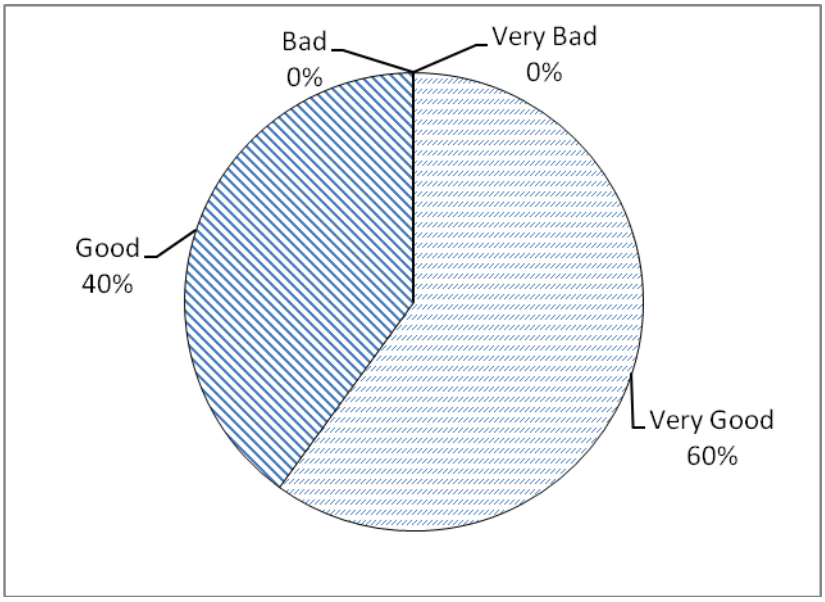


Figure 4.18: Idea of choosing, adding and configuring interface

17) Do you think other sites, for example, social networking sites should allow the users to select and arrange the items on their interface?

☐ YES ☐ NOT SURE ☐ NO

Why? Please provide a comment:

Comments provided by the participants are:

a) Participant who answered "No" said "*It makes it difficult to find things on other people's pages, because everyone positions things differently*".

b) Participant who answered "Not Sure" said "*It's nice to be able to customize for my own preferences but if everyone has their own customized display it may not have a*

standard look that people are use to. But if they can customize their view for only their own use then it is definetly useful".

c) Participant who answered "Yes" said "*So that users can choose what to display and what to follow*".

d) Participant who answered "Yes" said "*It would be good and fun for the users to have these features on Facebook*".

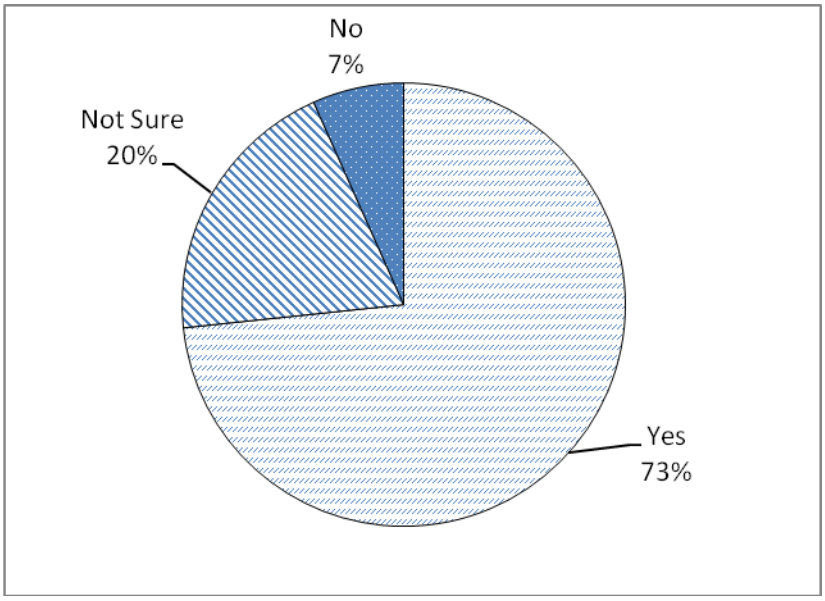


Figure 4.19: Should other social networking sites allow customization

18) Are there any other comments you would like to make about the systems or the study?

Why? Please provide a comment:

Comments provided by the participants are:

a) *Interesting and an [sic] user friendly system. I liked the social aspect of being able to exchange knowledge with others that I may know*.

b) *The idea of customizing one's community is interesting, but it's hard to synchronize all the users.*

c) *This idea is good. Adding more extra functionalities and features would be attractive like uploading photos, tags etc.*

d) *Make the users life comfortable with advanced technology. Keep trying, Good work.*

e) *It would be better to do the customization without repeatly checking these boxes .*

f) *When we click "Grouping", it will be nicer to give options to select the windows which we want to group (then, automatically fitting those windows in the grouping window). Because I found "dragging" function difficult.*

g) *It is very nice interface design. I enjoyed customizing my personal interface*

4.5.3 Observations

The application was tested on two different screens i.e. smaller and larger screens. When participants were asked which screen they liked the most, 87% of them chose the larger screen and 13% chose the smaller screen. One participant who chose the larger screen answered that he liked to work on both the screens.

After logging into the application, participants need to register into the communities that are provided for them. I have provided the option for the participants to add only one community at a time, but one participant pointed out that it would be good if there was an option to add all the needed communities at a time. The participants were able to see in their personal interface the communities that they were registered in and could not see other communities which were not added by them. One participant pointed out that it would be good if they could see other communities which they were not registered in.

After registering for a community, the participants added the functionalities that they thought were useful for them. When participants were more interested in a particular community they added all the functionalities of the community but for other communities which they were not much interested, they chose only a few functionalities. For adding functionalities to their personal interfaces 80% of the participants (twelve participants) used the Up Front strategy and only 20% (three participants) used As You Go strategy. Of the participants that used the As You Go strategy, they did it for different reasons: a) They wanted to group similar functionalities related to different communities together and did not find a functionality related to that particular community. b) Since different tasks were given to the participants while performing the experiments, sometimes participants lacked a particular functionality which was needed for the task, but which they had not added. c) They revisited the personal interface to check if there were any other functionalities which they did not add to personal interface.

After the functionalities appeared on the participants' personal interface, they arranged them according to their interests and preferences. The functionalities can be placed separately or can be grouped together. Two participants grouped functionalities to check if that particular customization option was working. One participant first tried to arrange functionalities on the personal interface separately but since the interface became cluttered, wanted to group the functionalities. From my observation I noticed that, 53.3% of the participants (eight participants) grouped functionalities based on the community. 20% of them (three participants) grouped based on the functionality and the other 26.6% (four participants) grouped unevenly i.e. grouped some functionalities and left some other functionalities separately or did not group at all. One user grouped few functionalities and minimized others, resized some others. One other participant minimized almost all functionalities that s/he added to the personal interface. One particular

participant resized functionalities so that all the functionalities were visible for him and did not group them (but first checked if that option was working and how it appears). Whenever this participant wanted to use that functionality, he maximized it and then restored it to the customized size once finished using the functionality and did not find any difficulty doing it always. One participant commented that it would be nice if the functionalities were grouped immediately after they clicked the "Group Items" button. This is not possible because the application does not know which functionalities users want to group and the grouping of items differs from one user to the other.

For arranging the functionalities on the participant's personal interface, I have provided the customization options for maximizing, minimizing and closing as checkbox options on a separate bar where the users need to select them, and the selections apply to all functionalities (boxes). But one participant pointed out that these options should be default options that should be present for every functionality box like the ones present for normal windows on a computer screen and did not want them as checkboxes which need to be selected.

Participants were asked to perform some tasks like uploading a file or link related to a particular community or interacting with other users in that community by answering a question or submitting a question after they are done customizing their community dashboard. But since the participants used the application for the first time, they spend most of the time customizing their community dashboard and only a small portion of the time performing the given tasks. Some users also deviated from the tasks and posted some other stuff which was not related to the community just to complete the task.

Participants liked the idea of dragging and dropping functionalities to a group and moving the functionalities from one place to another as a part of customization. While using the application users had some problems related to moving the functionalities on the screen and dragging-dropping to a group. These two customization options, moving and dragging-dropping, sometimes overlapped with each other and users were not able to drop the functionalities into a group. This was solved by users by just unchecking the "Allow drag" checkbox.

Two users pointed out that they needed a specific color for the accordion (a box which contains the grouped functionalities) also. One other user pointed out that customization options are useful for oneself in arranging the functionalities but if others visit our website, then they might have a problem identifying the functionalities which we have added and arrangements we have made. This is the case when users have the option to visit other users' dashboard to see the communities that they are registered for and the functionalities they have added. Currently this is not possible in my CMS. One user pointed out that the animation for grouping was good since I have used an accordion which makes the things go up and down.

The application was not designed for computer programmers but for any computer user. The customization options that were provided were also the ones that are used by any computer user on a daily basis - like maximizing a window, minimizing a window, closing, dragging the icons on the desktop, resizing a window to a particular size, dropping files into folders etc. Users with computer science background found it easy to start using the application immediately after logging in, but users from non-computer science background (2 out of 3) found it difficult to start using the application. They searched for different options and were testing all the options that were provided for them.

Figures 4.20 to 4.24 show the sample screen shots of customized by users community dashboards in FI. Some users have grouped the functionalities based on the community, some have grouped based on the functionality, and some of the functionalities were minimized and some other functionalities were restored to a particular size and were arranged on the interface.

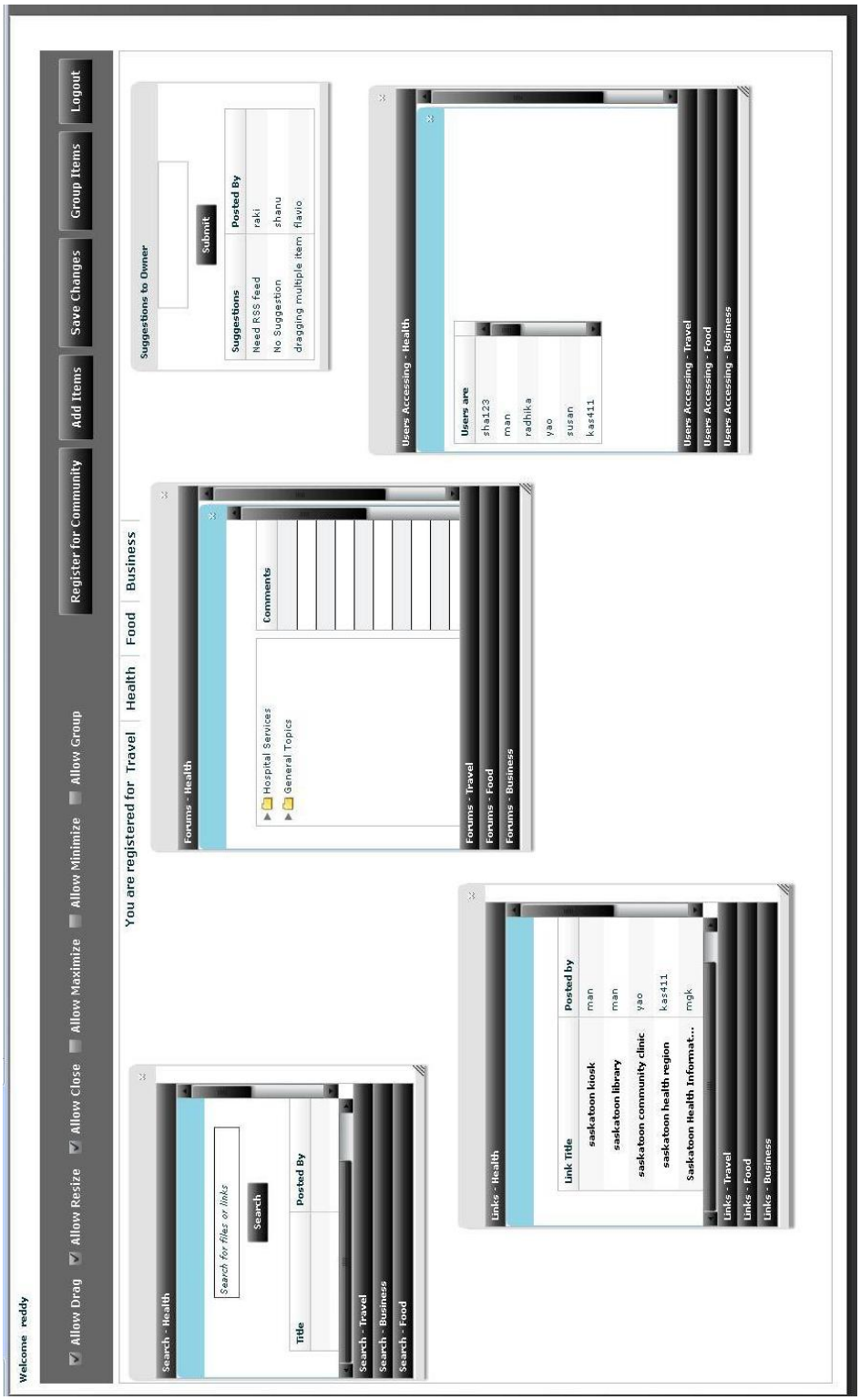


Figure 4.20: FI of a user who grouped functionalities based on similar functionalities present in different communities

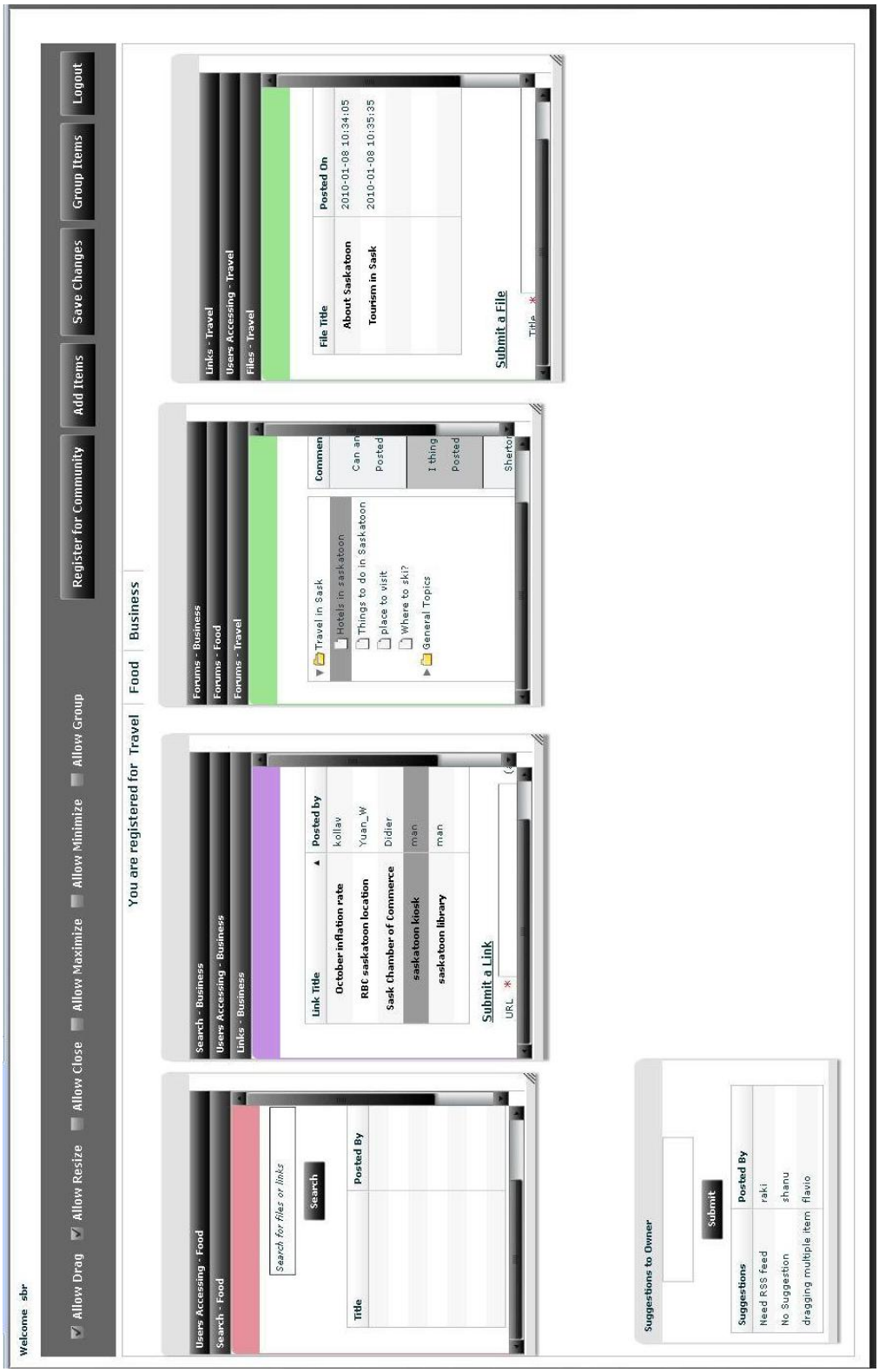


Figure 4.21: FI of a user who created 3 groups based on the community and the other based on the functionality

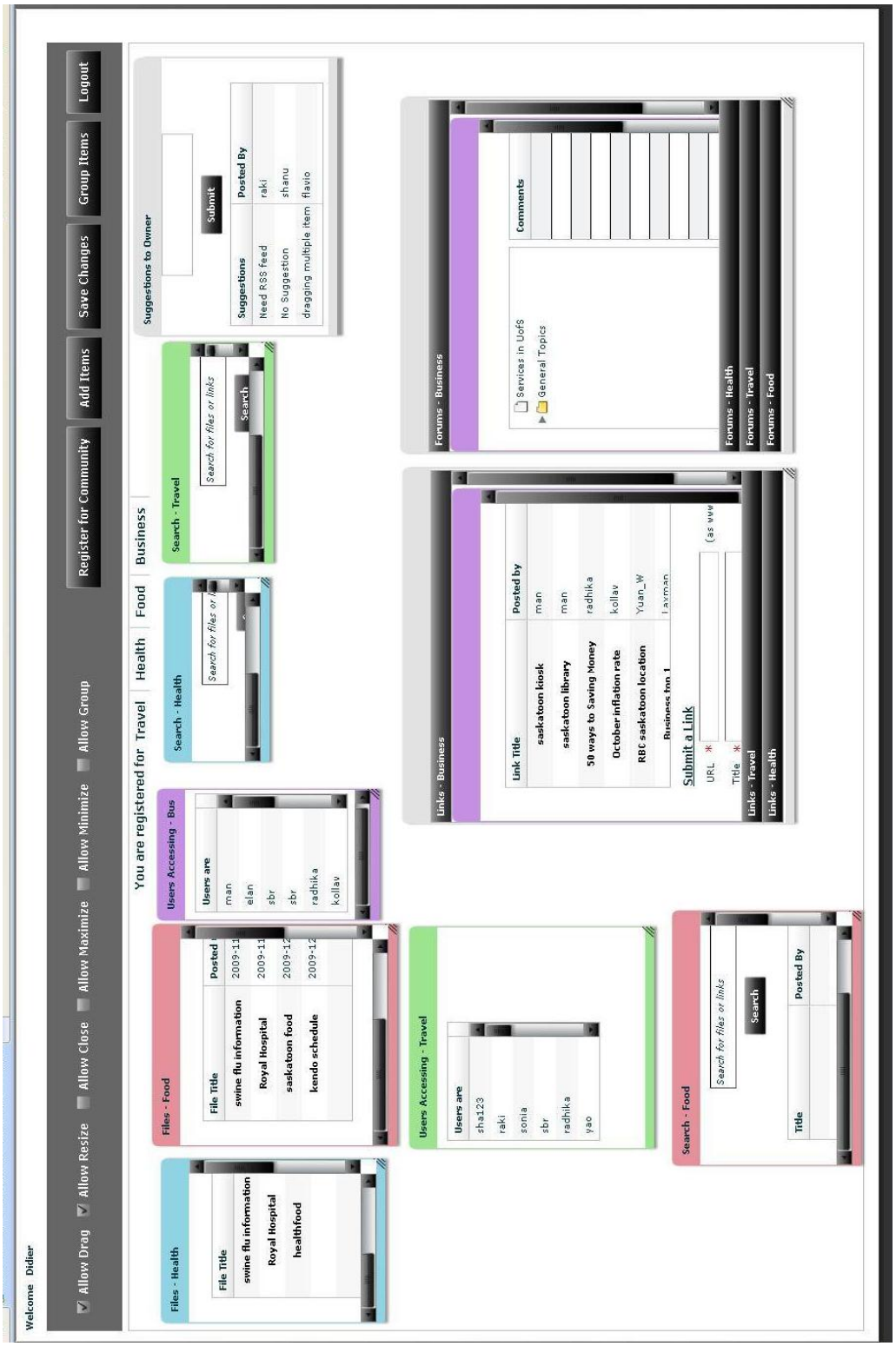


Figure 4.22: FI of a user who created two groups based on the functionality and placed the other functionalities separately by restoring them to a particular size

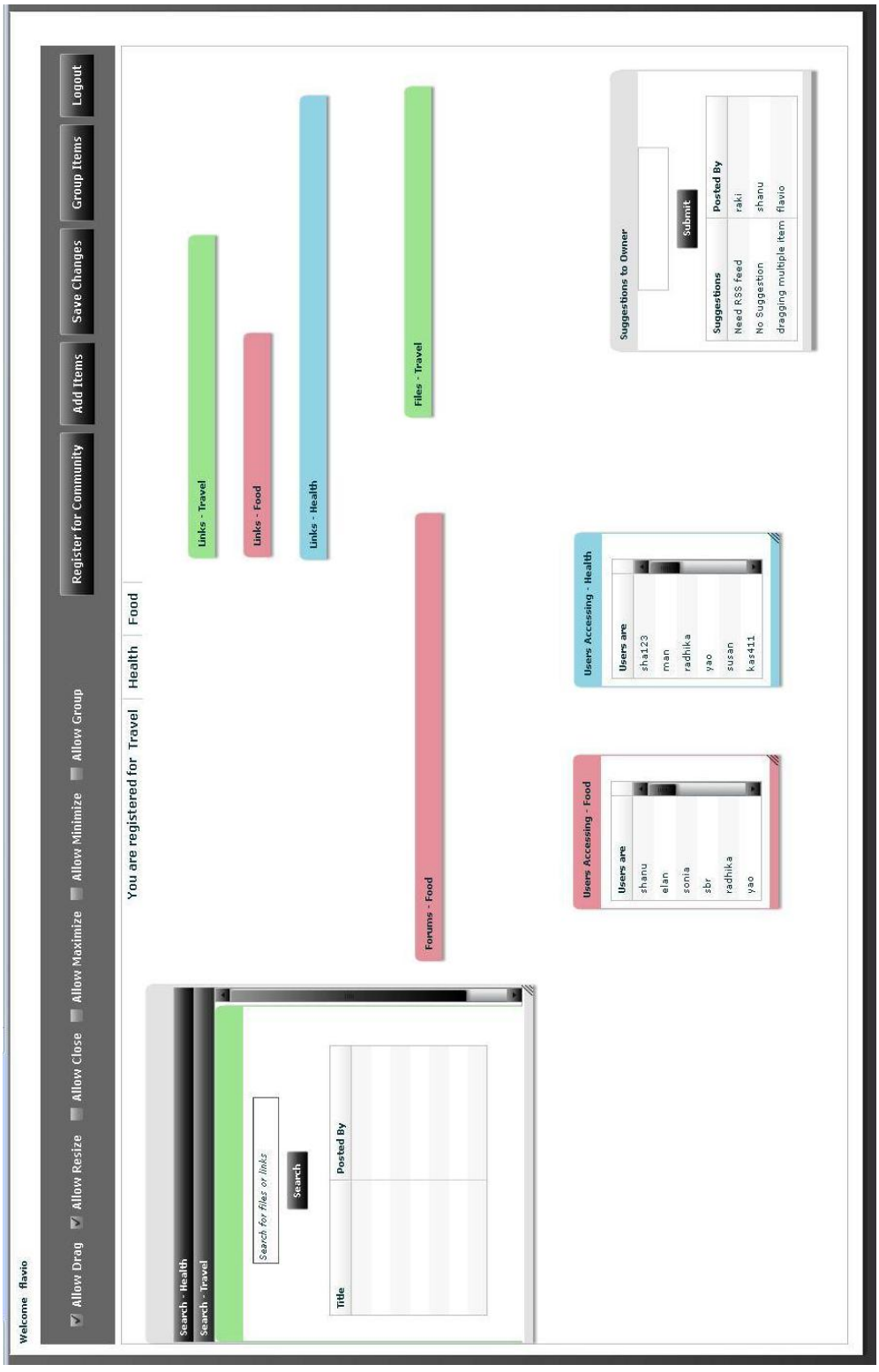


Figure 4.23: FI of a user who created one group based on the functionality and placed the other functionalities separately by minimizing most of them

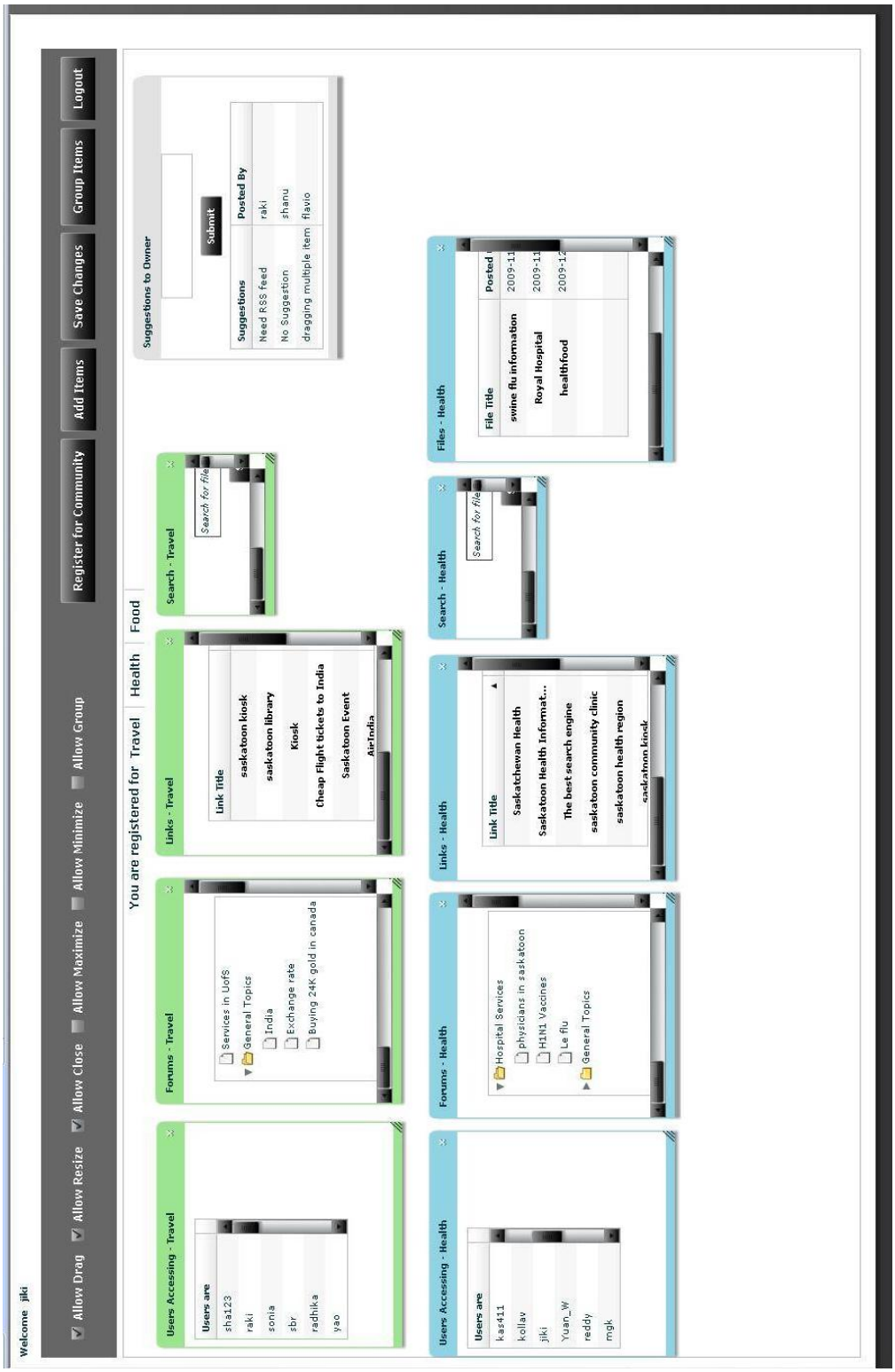


Figure 4.24: FI of a user who arranged functionalities by restoring them to a particular size

4.6 Limitations of the study

a. *Laboratory vs. Field Study:* The evaluation of the application was conducted in a laboratory with a group of users who are invited to work on the application for 45 minutes. An alternative for this would be to allow users to work on the application for a longer period of time, for example, in the context of a university class where they could share files and links related to the class.

b. *Deviation from tasks:* The users were given different tasks to be performed while using the application like uploading files, uploading links or posting discussion items in the forums related to the community. Users performed the tasks effectively but some of them deviated from the tasks and uploaded links which were not relevant to that particular community and which were done just to complete the task.

c. *Length of the experiment:* The length of the experiment was 45 minutes. We could have invited the users to explore their dashboards and the communities for several days. If they did, the results of the study would have been stronger, since it would have come close to a field study. However, they might have also not tried anything at all. Therefore, I didn't think that it is worthwhile engaging the users for a longer time experiment. However, a field study, by applying the proposed CMS to develop several communities where the users will be engaged and observing their customizations for a longer period of time is worthwhile (this would be a field study).

d. *Simplicity of the interface:* The communities that are present in the CMS have only a few functionalities and the application was easy to use with the customization options

that were provided. In real world, there will be communities which are complex with many functionalities and different types of goals to be done.

4.7 Discussion

The results of the evaluation study provide support for the stated hypotheses.

As stated in the "good idea" hypothesis I expected that the participants in the study would like the idea of using an online community with customization options allowing them to personalize their interface and functions to the community. The results obtained for Q4 and Q16 provide support for the good idea hypothesis.

The usage hypothesis stated the expectation that the participants would use the options to choose functionalities which they want and customize their community dashboard. The results obtained for Q3 and Q7 support the usage hypothesis since the majority of the participants stated that they added a subset of the available functionalities and all participants stated that they used the customization options for the interface. The results might have been skewed by some expectation effects since the participants knew that the goal of the experiment is to evaluate customization options and interface.

The satisfaction hypothesis stated that customization does not take much time and that the customization options that were provided were useful. Answers to questions Q6, Q8, and Q9 provided support for this hypothesis, by ranking highly the usefulness of most of the customization options (grouping, dragging & dropping and moving), by stating that customization does not take too much time and that the time investment is acceptable if the application is used frequently. One explanation was that the customization options used in this

application were fairly limited when compared to the complex customization options which might have been more time consuming. The overall reaction of the application was that it is easy to use (Q1) and the participants stated that it was interesting and user friendly application which would make a user's life comfortable with advanced technology (Q18).

The Navigation hypothesis stated the expectation that the participants would find it easy to navigate between the personalized (customized) and full interface for the community. Support for this hypothesis was provided by the participants' answers to Q11, where all the participants except for one agreed that it was easy to navigate between the PI and FI to add functionalities related to different communities because PI is a subset of FI and all participants said that it was easy to navigate and register for a community which they are interested in. However, 33% of the participants found that it was difficult to identify the hidden functionalities, once the customization is done. This is a potential problem, since users who have personalized away some functionalities may forget that they exist and become partially isolated from activities happening in the community.

While it was expected that users would perform more customizations in the smaller screen condition, the results did not confirm this expectation. When the participants were asked if space was a hurdle in arranging functionalities on a smaller screen in Q2 - 53% of them answered "Yes" and 47% of the answered "No", so there was no big difference in their perception. It is possible that the question was ambiguous, and the participants interpreted it in different ways. While arranging functionalities on a smaller screen could be harder than on a larger one, the ability to customize could be more useful exactly for the dashboard interface

viewed on a smaller screen, since it allows removing rarely used functions and reducing the clutter of the interface.

The decision to use color as a characteristic of the community in the customization options was good, as shown by the answers to question Q5, where nearly 90% of the users stated that it was useful to identify the functionalities related to different communities. Three participants requested the ability for changing the color of functionalities in response to Q15. From the answers it wasn't clear if they would have preferred to use colors for different functionalities rather than for different communities. This is certainly an option, but in combination with using color to distinguish different communities, it may lead to confusion. Adding the option to distinguish functionalities by another graphical feature of the boxes (e.g. texture) would allow integrating this suggestion in future versions of the system.

The majority (87%) participants liked the idea of combining different communities at a single place (Q10). This shows that there is a need of community dashboard applications that allow users to login into single application and combine functionalities from different communities at a single place.

40% of the participants said that they needed suggestions to do a particular type of customization because they thought that suggestions might bring new ideas for doing the customizations. In their comments, some users said that they need suggestions when they are using the application for the first time, but they also pointed out that suggestions at the later part of time when they are more experienced with the application might be frustrating.

The results of the study allowed better understanding of the reasons why these users do customizations and what patterns of customizations are preferred.

As the results in Figure 4.3 show, a different proportion of the available functionalities different communities were added by the participants. The least popular functionalities were the "*Files*" functionality (which allows users to share files) and it was chosen by the least number of users in all of the communities. Obviously, the majority of participants were not interested in sharing files with each other in all of the communities. This could possibly be due to the nature of the communities, discussing topics of general interest similar to many existing discussion forums or news sites. One could speculate that in communities with topics that are work or study related, e.g. in a community related to a class, users would be more interested in sharing documents, as they would be more likely to author documents. The "*Users accessing*" functionality was not chosen by the users of the health and food communities, so in these communities the majority of users would be unaware who is currently accessing that community. Other functionalities were popular in all communities. For example, "*Forums", "Links" and "Search*", were accessed by more than 50% of the users from all the communities. Since the majority of the users have access to these functionalities, these functionalities would be central for the interactions in the community. However, the few users who did not add these functionalities would be unaware of what is going on in that community related to that particular functionality, e.g. they would be unaware of the discussions happening in the community if they didn't add the "Forums" functionality in their personal interface. This can be a possible threat since it can lead to a fragmentation of the community. Yet, the users are autonomous and will add only those functionalities which are needed and that they think might be useful for them. One possible good application for adaptive intervention would be a notification service that alerts the user about community activities that are happening and the user may be unaware of, due to missing functionality as well as extra functionality.

All fifteen participants agreed that they customized the interface to complete the given tasks quickly (Q14). One possible reason is that the customization allows them to hide functions that are not used often and clutter the interface. Evidence for this is provided by the answers of nine participants (60%) who agreed that they customized to reduce the number of functionalities. The answers of additional seven participants (46%) who agreed that they customized to make the personal interface as small as possible can also be considered as an indication of the wish to simplify the interface. Users can also place the functionalities in such a way that they can have the most frequently used functionalities in a place which is more accessible to them.

Regarding the possibility for adding more customization options, most of the participants stated that they are satisfied with available options (Q15).

The users after doing the required customizations need to click the "save changes" button so that the changes are saved for future sessions. Two users suggested including a "reset" button which resets the customizations to the previously saved state when the users find the current customized versions inconvenient. This can be considered as probably a good suggestion that can be included in future versions of the application. Another user suggested dividing functionalities into tabs which is certainly an option that can also be pursued.

The majority of the participants (73%) agreed with the statement that other social networking sites should support this type of customization (Q17). Some participants commented that they can choose what they want to display and it would be good and fun for the users to use these options on any other social networking applications.

The results obtained in the study are certainly encouraging. However, they cannot be conclusive, since it was a small study conducted in a laboratory. The alternative for this would

have been conducting a field study by allowing users to use the application for particular amount of time in their own communities. As stated earlier, this would not have been feasible due to the need to create several thriving communities using the proposed CMS and evaluating it in a long period of actual use with users from these communities. But the laboratory study also provided some benefits related to the execution of the application. Since it was the first time the users were using the application and there were no customization suggestions provided to the users (I chose the non adaptive approach), the users needed some guidance from me in using the application from time to time (even though instructions were given to them at the beginning). The users were asked to think-a-loud while using the application and I took notes, so it was easy for them to ask questions, as I was sitting next to them in the lab. This method did not contaminate the results as I was just clarifying their doubts regarding the application and the functionalities that are present.

The time duration was normal for a lab experiment. However, in reality users access online communities regularly in long periods of time, e.g. weeks, months, or even years, and their way of interaction with the communities change as they learn about them. The users might have had different views if they had used their customized dashboards to access their communities for a longer period of time, e.g. several days or weeks.

In this study, the participants performed the customization at the beginning of their use of their community dashboards. In a field study, where the users would have been involved in a long term interaction with their communities, participants might have used a different strategy, such as “Customize As You Go”, as it has been shown in (McGrenere et al, 2002). In our case,

since the application was not embedded in their normal daily activities, most of the users used "Up Front" strategy.

Despite these limitations, this evaluation is a first step showing that customizations in the user's access points to online communities will help users to complete their tasks more easily (Q14). It also shows that users are open to the possibility of combining functionalities in different communities at a single place in a community dashboard. Work by another student at the MADMUC Lab (Wang, Zhang, and Vassileva, 2010) follows up on these directions by creating personal dashboard for accessing different social networking sites (FaceBook and Twitter).

4.8 Summary

This chapter presents the evaluation and results of CMS, with two interfaces for testing the customization strategies on online communities. The hypothesis were good idea hypothesis, usage hypothesis, satisfaction hypothesis, screen-size hypothesis and navigation hypothesis which are related to the main questions of whether is it a good idea to customize the functionalities, do users actually use the customization options, are users satisfied with the customization options that are provided in the community or are users able to navigate between the personal interface and full interface. To test the hypothesis users were asked to use the CMS with customization options and design their personal community dashboards. The results showed that users used the customization options for designing the community dashboard and are satisfied with the customization options that are provided. The users were able to navigate between the full interface and personal interface to access the functionalities and they think that it is a good idea to be able to customize the functionalities of an online community. The

expectation of the screen-size hypothesis was not met as participants did not think that the space was hurdle in the application with smaller interface. The results provided the evidence for supporting the hypothesis, however, to be able to claim with certainty that these hypothesis hold, more studies are needed, in authentic environment, with more and more diverse users.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

The goal of this thesis is to address a weakness existing in current CMSs and design a CMS which allows end-users along with the owner of the community to make customizations to their personal interfaces to the online communities. The proposed CMS allows end users to configure the functionality of the community as a subset of the full interface created by the community owner without using any programming knowledge. From the user's point of view, the advantage is that it is easy and simple to do customization. From the designer's point of view, the advantage is that there is no need to make any final decisions regarding the functionality and design of the application to satisfy the majority of users. This chapter summarizes the contributions that were made in this thesis, followed by the future research directions that can follow.

5.1. Contributions

5.1.1 Created a CMS Framework with Flex for creating many communities

Online communities used within organizations are most often developed using CMSs. All existing CMSs are developed with popular programming languages such as Java, .Net, PHP, Perl etc., but there are no CMSs developed using Adobe Flex. In this research the CMS was developed using Adobe Flex on the client side and PHP on the server side. The advantage of using Adobe Flex is the ease of development of client-side web interfaces that have an excellent performance from user's point of view and work on all existing browsers.

The proposed CMS allows registered users to create communities and become the owners of their communities. The owner of a community needs to choose the full set of possible

functionalities that would be available to the members. Users who want to become members of a community need to register in the community. They will get access to all the functionalities that are provided by the owner of that community.

5.1.2 Allow end-users to customize their personal interface to the online community

Currently there are no CMSs that allow end-users to customize the community interface by adding the functionalities which they want on their community's homepage. Such customization exists only in Web portals, such as My Yahoo!, and iGoogle where the user can choose and arrange the functionalities which they want. However these personalized portals do not allow users to resize the functionalities to a particular size or to group the functionalities into a single box. Existing CMSs allow community owners (who are also the community designers and possess programming skills) to choose the functionalities and design their community interface using a chosen predefined template. Later, all users of that community will use the application as it was designed by the owner. The proposed CMS allows end users along with the owners of the community to customize the interface and arrange the functionalities on their personal dashboard according to their preferences. After registering in the community, the users are able to choose the functionalities which they want to use and add them to their personal community interface (dashboard). Once they have added functionalities, by using the customization options (moving, resizing, dragging, dropping and grouping) the users can arrange them on the screen according to their preference and save their configurations for future use. Each and every user has their own goal in using the application and will be customizing the application in their own way depending on their goals. This CMS will help users to achieve their goals easily by choosing and customizing the functionalities rather than using the full interface provided by the community.

5.1.3 Allow users to create a personalised dashboard to view different communities simultaneously

Typically, CMS allow creating online communities, where users log in and participate in separately, one community at a time. In contrast, the MUCCD CMS allows users to create a personalized dashboard for viewing the functionalities related to several different communities simultaneously. Users can register for any number of communities that are available to them and add the functionalities related to those communities at one place - their personal dashboard. In this way they can access conveniently the functionalities related to those communities at the same place without the need to open new windows, log in, etc. Users also have an option to group the functionalities either by community or by the functionality type. An example of this is shown in the thesis (Chapter 4) where the participants in the evaluation study registered for communities travel, health and food could choose the functionalities and create their personalized dashboard's with the functionalities related to all the communities that they were registered. By using the customization options, users can arrange functionalities on the interface and can group functionalities depending on their preferences for visual layout/organisation. To our best knowledge this is a novel type of integrative application for online communities and social networks and there is very little work in this area. Of course, a limitation of our approach is that all communities / social networks must be created within our CMS; it is impossible to link to communities / social networks that exist outside.

5.2 Future Work

5.2.1 Extending the customization options to a finer grain level

In the MUCCD CMS customizations can be done only on functionalities that have been created for the community by the owner. These customizations affect the choice of a subset of functionalities and the proposed layout (partition, size, and grouping). It would be good to have the ability to customize the content of the functionalities. For example, the functionality "Links" provides a list of all links shared in the community and these could be sorted in different ways - by time of sharing, by number of views, by rating. The choice of a sorting criterion for the shared links is an option of the particular functionality, which cannot be customized currently, but it would be very good to be able to do so. Choosing a "default view" of the content presented by various functionalities is another example of a finer-grain customization, on the level of a particular functionality that can be pursued in the future.

5.2.2 Adding an adaptive approach for recommending useful customization

An adaptive approach could be added to the application, by creating a user modeling component that keeps track of user actions and gives users suggestions in arranging the functionalities or adding other functionalities. Users can also get suggestions to add a particular functionality which they have removed, but where a lot of community activity is going on, to make sure they are not missing out on important activities going on in the community. To achieve this, a model of community activity will have to be developed, which is currently an exciting new area in User Modeling and Personalization. Another kind of adaptivity would be to discover certain kinds of customization that are very efficient for a particular community and to suggest this customization to the users of similar communities. This would be a community-customization, rather than a personal customization.

5.2.3 Changing color

In the MUCCD CMS, different colors are assigned to different communities so that they are easy to identify, when there are more than one community's in the user's dashboard. It could be a good idea to allow users to assign different colors or patterns (e.g. checkered, striped., etc.) also to the different functionalities, so that they can distinguish them based on their preferences and make them easier to locate on the screen. Along the same lines, assigning colors or textures to different groups can also allow users to identify the groups easily.

5.2.4 Share customizations between users

Each user may arrange the functionalities according to their interest and preferences in using the application. It may be helpful for users to know how others have arranged their dashboards, since they may find better customizations than their own and may want to change their customizations accordingly. For this, users should be given a chance to visit the dashboards of other users and view their customizations. However, there might be some problems with this type of approach as pointed out be one participant in the study, since the users visiting other user's dashboard may not understand the customizations that are done by that particular user. There may also be privacy problems involved that will have to be considered.

5.2.5 Conduct a field study

The evaluation of the proposed CMS was done in a laboratory setting with a limited number of participants. In order to obtain more accurate results and to know how users do their customizations for their real communities, the users should be given a chance to work with the application for a longer period of time in communities that are really relevant to their work or study. For this more functionalities will be added depending the context of study that will be

conducted. In the current evaluation study only the screen size was considered a factor that may force the users to customize the application. But the purpose of use can also influence the users to customize the application i.e. if two users agree to work together and want to interact actively, then they both can agree to customize the application in a particular way.

By conducting a field study in different areas with the people of different ages and different levels of computer knowledge we can discover what factors influence the users to make customizations and what type of customizations will be done by users. For example, people who have less experience with the computer may add only the functionalities which they think would need and other users may add the functionalities either just because of their interest or to design the application in a pleasing manner.

5.2.6 Creating dashboard applications for heterogeneous online communities and social network sites

Currently, the proposed approach allows users to access and personalize only the homepages of communities created using my MUCCD CMS. A future direction that would be worthwhile for this approach to gain traction is to provide a dashboard that allows users to access and customize their view of many existing online communities in which they participate. Research in this direction is already underway in the MADMUC Lab (Wang et al, 2010) where a dashboard for accessing different social networking sites (Facebook and Twitter currently) has been created, which allows not only GUI adaptation, but merging content and friends across different social networks.

5.2.7 Version Management

Users should be able to create different versions of the application depending on the context of their use and depending on the purpose. For example a user can create one particular type of customization when working individually on the application and can create another type of customization when working on a group project. She/he should easily be able to switch between the versions were created. Users should be able to save particular customizations under a name and retrieve them later.

5.3 Conclusion

The contribution of this thesis is the design and implementation of a Flex-based CMS that allows customization of the community interface by end-users. This CMS allows users to create their personal interfaces to the communities they participate in without using any programming knowledge. The personal interfaces can contain subsets of the original functionality provided for the community by its owner/developer, including only those functions that the users really use, and the functionalities can be arranged according to the user's preferences for their convenience. The evaluation of this approach in a lab study indicate that users like the idea of customizing their community dashboard and are able to perform customizations and are satisfied with the interface created for this purpose.

I foresee that the dashboard applications where users can adapt their online community interfaces will gain importance with the proliferation of online communities, and social networks as well as small devices (netbooks, smart phones).

REFERENCES

Arakelyan, A. (2009, March 28). Introduction to web content management systems. Retrieved on September 1, 2009, from Slideshare. http://www.slideshare.net/formativesystems.com/introduction-to-content-management-systems-cms-1292337.

Benyon, D. (1993). Accomodating individual differences through an adaptive user interface. In *Adaptive User Interfaces – Results and Prospects.* M. Schneider-Hufschmidt, T. Kuhme, and U. Malinowski. (eds.), Elsevier Science Publishers, Amsterdam, 149–165.

Bunt, A. (2007). Mixed-initiative support for customizing graphical user interfaces. Doctoral Dissertation, University of British Columbia, Vancouver, Canada.

Bunt, A., Conati, C. and McGrenere, J. (2007). Supporting interface customization using a mixed-initiative approach. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*. IUI'07. ACM, New York, NY, 92-101.

Bunt, A., Conati, C., and McGrenere, J. (2004). What role can adaptive support play in an adaptable system?. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*. IUI '04. ACM, New York, NY, 117-124.

Bunt, A., Conati, C., Huggett, M., and Muldner, K. (2001). On improving the effectiveness of open learning environments through tailored support for exploration. In *Proceedings of AIED 2001, 10th International Conference on Artificial Intelligence in Education*. San Antonio, TX, U.S.A, 365–376.

Carroll, J., and Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors* 26, 377–389.

Chen, L., and Pu, P. (2007). Hybrid critiquing-based recommender systems. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*. IUI' 07. ACM, New York, NY, 22–31.

Christiernin, L.G., Lindahl, F., and Torgersson, O. (2004). Designing a multi-layered image viewer. In *Proceedings of the Third Nordic Conference on Human-Computer Interaction*. NordiCHI'04 82. ACM, New York, NY, 181–184.

Clark, B., and Matthews, J. (2005). Deciding layers: Adaptive composition of layers in a multi-layer user interfaces. In *Proceedings of 11th International Conference on Human-Computer Interaction* 7.

Corlan, M. (2010, March 22). The architecture of Flex and PHP applications. Retrieved April 22, 2010 from Adobe Developer Connection. http://192.150.14.31/devnet/flex/articles/flex_php_architecture.html.

Cote-Munoz, J. A. (1993). AIDA—An adaptive system for interactive drafting of CAD applications. In *Adaptive User Interfaces: Principles and Practice*. M. Schneider-Hufschmidt, T. Kuhne, and U. Malinowski. (eds.), Elsevier Science Publishers, Amsterdam.

Cypher, A. (1991). Eager: Programming repetitive tasks by example. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. CHI'91. 33–39.

Debevc, M., Meyer, B., Donlagic, D., and Svecko, R. (1996). Design and evaluation of an adaptive icon toolbar. *User Modeling and User-Adapted Interaction* 6, 1–21.

Dieterich, H., Malinowski, U., Kühme, T., and Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. In *Adaptive User Interfaces: Principles and Practice*. M. Schneider Hufschmidt, T. Kuhme, and U. Malinowski. (eds.), Elsevier Science Publishers, Amsterdam, 13–48.

Edutech Wiki (2010, March 2). Learning management system. Retrieved on April 24, 2010. http://edutechwiki.unige.ch/en/Learning_management_system

Ellis, R (2009), A Field Guide to Learning Management Systems, ASTD Learning Circuits. Retrieved on April 24, 2010. http://www.astd.org/NR/rdonlyres/12ECDB99-3B91-403E-9B15-7E597444645D/23395/LMS_fieldguide_20091.pdf.

Fenton, K. (2008). What is a Content Management System, Do I Need One? Retrieved on September 1, 2009, from Nextstep design solutions. http://www.nextstep-designsolutions.com/index.php?option=com_content&view=article&id=82:what-is-a-content-management-system-do-i-need-one&catid=37:news-marketing.

Findlater, L., and McGrenere, J. (2004). A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI'04. ACM, New York, NY, 89–96.

Findlater, L., and McGrenere, J. (2007). Evaluating reduced-functionality interfaces according to feature findability and awareness. In *Proceedings of INTERACT*, 592-605.

Fischer, G. (1993). Shared knowledge in cooperative problem-solving systems—integrating adaptive and adaptable components. In *Adaptive User Interfaces: Principles and Practice*, M. Schneider-Hufschmidt, T. Kuhme, and U. Malinowski. (eds.), Elsevier Science Publishers, Amsterdam, 49–68.

Fischer, G. (2001). User modeling in human-computer interaction. *User Modeling and User-Adapted Interaction* 11, 65-86.

Flex 3 benefits. (n.d.) Retrieved July 19, 2009, http://www.adobe.com/products/flex/features/flex3/.

Flex Overview. (n.d.). Retrieved July 19, 2009, http://www.adobe.com/products/flex/overview/.

Gajos, K., Czerwinski, M., Tan, D. S., and Weld, D. S. (2006). Exploring the Design Space for Adaptive Graphical User Interfaces. In *Proc of AVI*, 201-208.

Gant, M., and Nardi, B.A. (1992). Gardeners and gurus: Patterns of cooperation among CAD users. In *Proceedings of the ACM Conference on Human Factors in Computing Systems.* CHI'92. 107–117.

Gong, G., and Salvendy, G. (1995). An approach to the design of a skill adaptive interface. *International Journal of Human-Computer Interaction* 7, 365–383.

Greenberg, S., and Witten, I. (1985). Adaptive personalized interfaces—A question of viability. *Behaviour and Information Technology* 4, 31–45.

Hinchcliffe, D. (2008, September 4). Ten leading platforms for creating online communities. Retrieved on September 1, 2009 from ZDNet News & Blogs. http://blogs.zdnet.com/Hinchcliffe/?p=195.

Horvitz, E., Breese, J., and Henrion, M. (1988). Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning* 2, 247–302.

Horvitz, E., Herckerman, D., Hovel, D., and Rommelse, R. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. UAI'98. 256–265.

Jameson, A., and Schwarzkopf, E. (2002). Pros and cons of controllability: An empirical study. In Proceeding of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, 193–202.

Jorgensen, A.H., and Sauer, A. (1990). The personal touch: A study of users' customization practice. In *Proceedings of INTERACT*, 561–565.

Karen (2009, January 5). Drupal Learning Curve. Retrieved February 9, 2010 from Library Web Chic. http://www.librarywebchic.net/wordpress/2009/01/05/drupal-learning-curve/.

Kieras, D.E., Wood, S.D., Abotel, K., and Hornof, A.J. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. UIST'95. 91–100.

Krogsaeter, M., Oppermann, R., and Thomas, C. (1994). A user interface integrating adaptability and adaptivity. In *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*, Lawrence Erlbaum Associates, Hillsdale, NJ, 97–125.

Mackay, W. E. (1991). Triggers and barriers to customizing software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology*. S. P. Robertson, G. M. Olson, and J. S. Olson. (eds.), CHI'91. ACM, New York, NY, 153-160.

MacLean, A., Carter, K., Lovstrand, L., and Moran, T. (1990). User-tailorable systems: Pressing the issues with buttons. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. CHI'90. 175–182.

Malinowski, U. (1993). Adjusting the presentation of forms to users' behaviour. In *Proceedings of the ACM Conference on Intelligent User Interfaces*. IUI'93. 247–249.

Manber, U., Patel, A., and Robison, J. (2000). Experience with personalization on Yahoo! *Communications of the ACM* 43, 35 - 39.

McGrenere, J. (2002). The Design and Evaluation of Multiple Interfaces: A Solution for Complex Software. Doctoral Dissertation, University of Toronto, Toronto, Canada.

McGrenere, J., and Moore, G. (2000). Are we all in the same "bloat"? In *Proceedings of Graphics Interface*, 187–196.

McGrenere, J., Baecker, R. M., and Booth, K. S. (2007). A field evaluation of an adaptable two-interface design for feature-rich software. In *ACM Transactions on Computer-Human Interaction* 14, TOCHI'07.

McGrenere, J., Baecker, R. M., and Booth, K.S. (2002). An evaluation of a multiple interface design solution for bloated software. In *Proceedings of the ACM Conference on Human Factors in Computing Systems.* CHI'02. 163–170.

Miah,T., Karageorgou, M., and Knott. R.P. (1997). Adaptive toolbars: An architectural overview. In *Proceedings of the 3rd ERCIM Workshop on User Interfaces for Al*.

Miniwatts Marketing Group (2010, April 17) Internet Usage Statistics. Retrieved April 24, 2010 from Internet World Stats. http://www.internetworldstats.com/stats.htm.

Myers, B., Hudson, S., and Pausch, R. (2000). Past, Present, and Future of User Interface Software Tools. In *ACM Transactions on Computer-Human Interaction* 7, 3-28.

Norman, K.L. (1991). The Psychology of Menu Selection. Ablex Publishing Corporation, Norwood, New Jersey.

Oppermann, R. (1994). Adaptively supported adaptability. *International Journal of Human-Computer Studies* 40, 455-472.

Page, S. R., Johnsgard, T. J., Albert, U., and Allen, C. D. (1996). User customization of a word processor. In *Proceedings of ACM*. CHI'96. ACM, New York, 340–346.

Papanikolaou, K. A., Grigoriadou, M., Kornilakis, H., and Magoulas, G.D. (2003). Personalizing the interaction in a web-based educational hypermedia system: the case of INSPIRE. In *User Modeling and User-Adapted Interaction* 13, 213–267.

Prayas (2008, February 24). Jargons. Retrieved September 1, 2009 from Information Corner. http://prayas4u.wordpress.com/jargons/.

Robertson, J. (2003, June 3). So, What is a CMS? Retrieved September 1, 2009, from Step Two Designs. http://www.steptwo.com.au/papers/kmc_what/index.html.

Sears, A., and Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. In *ACM Transactions on Computer-Human Interaction* 1, 27–51.

Shannon, M (2009, April 15). Open Source vs Proprietary Content Management Systems. Retrieved on April 22, 2010 from BestRank. http://www.bestrank.com/blog/open-source-vs-proprietary-content-management-systems-cms.

Shneiderman, B. (1997). Designing the User Interface: Strategies for Effective Human-Computer Interaction (3rd ed.). Addison-Wesley Publishing, Reading, MA.

Shneiderman, B. (2003). Promoting universal usability with multi-layered interface design. In *Proceedings of Conference on Universal Usability*. CUU'03. 1–8.

Stuerzlinger, W., Chapuis, O., Phillips, D., and Roussel, N. (2007). User interface facades: Towards fully adaptable user interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. UIST'07. 309–318.

Thomas, C.G., and Krogsoeter, M. (1993). An adaptive environment for the user interface of excel. In *Proceedings of the ACM Conference on Intelligent User Interfaces*. IUI'93. 123–130.

Tradocaj (2009, October). Advantages and Disadvantage of Joomla. Retrieved February 9, 2010 from SlideShare. http://www.slideshare.net/tradocaj/advantages-and-disadvantages-of-joomla.

Vassileva, J. (1996). A Task-Centered Approach for User Modeling in a Hypermedia Office Documentation System. In *User Modeling and User Adapted Interaction* **6**, 185-223.

Wang, Y., Zhang, J., and Vassileva, J. (2010). SoCConnect: A User-Centric Approach for Social Networking Sites Integration. *In Proceedings of the International Conference on Intelligent User Interfaces (IUI) Workshop on User Data Interoperability in the Social Web*.

Wikipedia (2009, August 24). Telligent Community. Retrieved August 29, 2009. http://en.wikipedia.org/wiki/Telligent_Community.

Wikipedia (2009, July 20). PHP. Retrieved August 20, 2009. http://en.wikipedia.org/wiki/PHP.

Wikipedia (2010, April 21). Autocomplete. Retrieved April 24, 2010. http://en.wikipedia.org/wiki/Autocomplete.

Wikipedia (2010, April 22). Learning Management Systems. Retrieved April 24, 2010. http://en.wikipedia.org/wiki/Learning_management_system.

Wikipedia (2010, April 24). Moodle. Retrieved April 24, 2010. http://en.wikipedia.org/wiki/Moodle.

Wikipedia (2010, April 24). WebCT. Retrieved April 24, 2010. http://en.wikipedia.org/wiki/WebCT.

Williams, M. (n.d.). Glossary. Retrieved on September 2, 2009 from Public Life. http://www.publiclife.co.uk/glossary.html.

Zhou, M.X., Wen, Z., and Aggarwal, V. (2005). A graph-matching approach to dynamic media allocation in intelligent multimedia interfaces. In *Proceedings of the ACM Conference on Intelligent User Interfaces*. IUI'05. 114–121.

APPENDICES

Appendix A – Consent Form

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF SASKATCHEWAN

You are invited to participate in a study entitled "Designing User Configurable Online Community Framework." Please read this form carefully, and feel free to ask questions you might have.

Researchers: **Julita Vassileva**, Department of Computer Science (966-2073), jiv@cs.usask.ca

Manju Chava, Department of Computer Science

The purpose of the study is to evaluate the effectiveness of use of the provided adaptable interface. In the study, you will be asked to register for one or more communities that are provided and add the functionalities which ever you think would be useful for that community and arrange all the functionalities according to your wish by dragging, dropping, resizing, grouping and changing the color. In this study you will answer some questions about your experience with the system. The estimate of the total time to participate in this study is 45 minutes.

There are no known risks in this study.

Findings from the study will be used to enhance the features of the system. You will be observed during your use of the community, and then you will need to fill a questionnaire. You will receive a $10 honorarium for your participation in the study.

The research data will be stored on a password-protected computer system and will be available only to the investigators. Identifying information will be destroyed upon completion of data collection, and then pseudonyms will be used to refer to the participants. All data will be kept by Dr. Vassileva for a minimum of five years upon the completion of this study. If Dr. Vassileva chooses to destroy the data after the five years, the data will be destroyed beyond recovery.

Aggregate results will be used in a thesis and articles published in conferences and journals. Any information that can be linked to a specific participant will be removed or altered.

Your participation is voluntary, and you may withdraw from the study for any reason, at any time, without penalty of any sort. You may refuse to answer individual questions. If you withdraw from the study, data that you have contributed will be destroyed at your request.

If you have any questions concerning the study, please feel free to ask at any point; you are also free to contact the researchers at the numbers provided above if you have questions at a later time. This study has been approved on ethical grounds by the University of Saskatchewan Behavioural Research Ethics Board on (insert date). Any questions regarding your rights as a participant may be addressed to that committee through the Ethics Office (966-2084). Out of town participants may call collect. You may find out about the results of the study through the MADMUC website (http://madmuc.usask.ca/) or by contacting the researchers.

I have read and understood the description provided above; I have been provided with an opportunity to ask questions and my questions have been answered satisfactorily. I consent to participate in the study that is described above, understanding that I may withdraw this consent at any time. A copy of this consent form has been given to me for my records.

_____	_____
(Name of Participant)	(Date)
_____	_____
(Signature of Participant)	(Signature of Researcher)

Appendix B – First questionnaire

Personal information (This information will be treated confidentially)

1. Age ☐

2. Sex ☐ Female ☐ Male

3. On a scale of 1 (beginner) to 5 (expert), how would you rate your computer skills as an end-user?

☐

4. How many hours do you browse the Web daily (on average)?

☐ Less than 1 hr ☐ 1-5 hrs ☐ 6-10 hrs ☐ More than 10 hrs

5. On a scale of 1 (beginner) to 5 (expert), how would you rate your general knowledge of the following topics?

- Travel: ☐
- Health: ☐
- Business: ☐
- Food: ☐

6. How many social community sites are you registered in?

☐ 0-2 ☐ 3-5 ☐ 6-8 ☐ Over 8

Please mention them here:

Appendix C - Pilot Study Questionnaire

This appendix presents a sample questionnaire for the pilot study of usability evaluation study.

1. Overall reactions to the system:

Terrible	1	2	3	4	5	6	7	8	9	Wonderful
Difficult to use	1	2	3	4	5	6	7	8	9	Easy to use
Dull	1	2	3	4	5	6	7	8	9	Stimulating
Frustrating	1	2	3	4	5	6	7	8	9	Satisfying
Overwhelming	1	2	3	4	5	6	7	8	9	Manageable
Complex	1	2	3	4	5	6	7	8	9	Simple

2. Did you add all the functionalities that were available to your personal interface?

   ○ **Yes** ○ **Maybe** ○ **No**

   Why? Please provide a comment:

3. Would you like that the system makes suggestions for you to do a customization which might be useful for you?

   ○ **Yes** ○ **Maybe** ○ **No**

   Comment:

4. Do you think other sites, for example, social networking sites should allow the users to select and arrange the items on their interface?

   ○ **Yes** ○ **Maybe** ○ **No**

   Comment:

5. Did you like the idea of combining different communities at a single place? (By "different communities" we mean groups discussing different topics, e.g. health, travel, or different classes.)

○ **Yes** ○ **Maybe** ○ **No**

Comment:

6. Were the following customization features useful for you in arranging the items? (Please tick one box for each question)

	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
a. Moving					
b. Resizing					
c. Dragging					
d. Dropping					
e. Grouping					

7. Please, rank these features, according to their usefulness to you:

- Moving -
- Resizing -
- Dragging -
- Dropping -
- Grouping -

8. Do you think arranging all items takes lot of time?

○ **Yes** ○ **Maybe** ○ **No**

Comment:

9. Generally, what do you think of the idea of choosing, adding and configuring your personal interface of the community?

○ **Very Good** ○ **Good** ○ **Bad** ○ **Very Bad**